COMMUNICATION WITH PLC'S WITH THE USE OF OPC TOOLBOX

L. Mrafko, M. Kajan, E. Viszus, V. Vodička

Institute of Robotics and Cybernetics, Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology

Abstract

The aim of this paper is to test the possibilities of OPC communication using two different types of PLCs (Programmable Logic Controllers) on the one side and Matlab OPC Toolbox on the other side. We try to measure transfer times of large quantities of data using different data types and different approaches.

1 Description of the test configuration

We are using two types of PLCs to compare OPC communication performance of both of them using the same approaches. The PLCs are connected together and to the PC equipped with OPC server via industrial-grade ethernet switch. The configuration is as follows and depicted in the figure 1:

PLC 1: Simatic S7 CPU 317-2 PN/DP (6ES7 317-2EK14-0AB0)

PLC 2: Simatic S7 CPU 412-2 DP (6ES7 412-2XJ05-0AB0) with communication module Simatic S7 CP 443-1 Advanced (6GK7 443-1GX20-0XE0)

Switch: SIMATIC NET Industrial Ethernet Switch SCALANCE X204IRT (6GK5 204-0BA00-2BA3)

PC: Simatic rack PC, Intel(R) Core(TM) 2 Quad CPU Q9400 @ 2,66 GHz, 3GB RAM, Simatic NET OPC server Edition 2006, Matlab R2010b, OPC Toolbox.

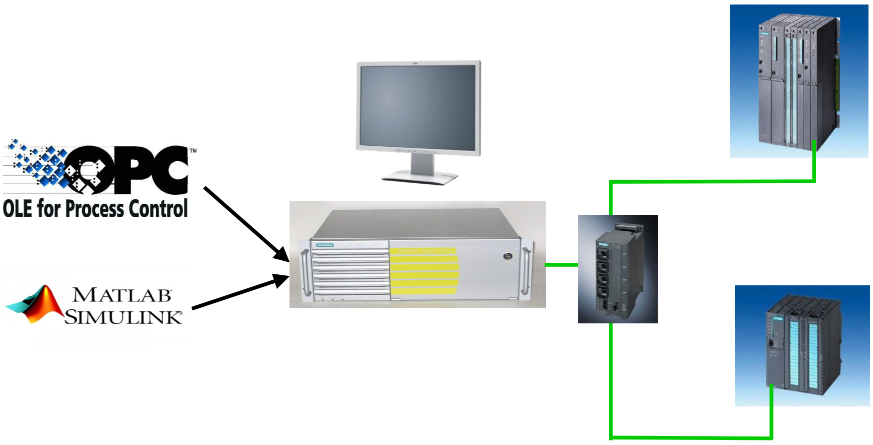
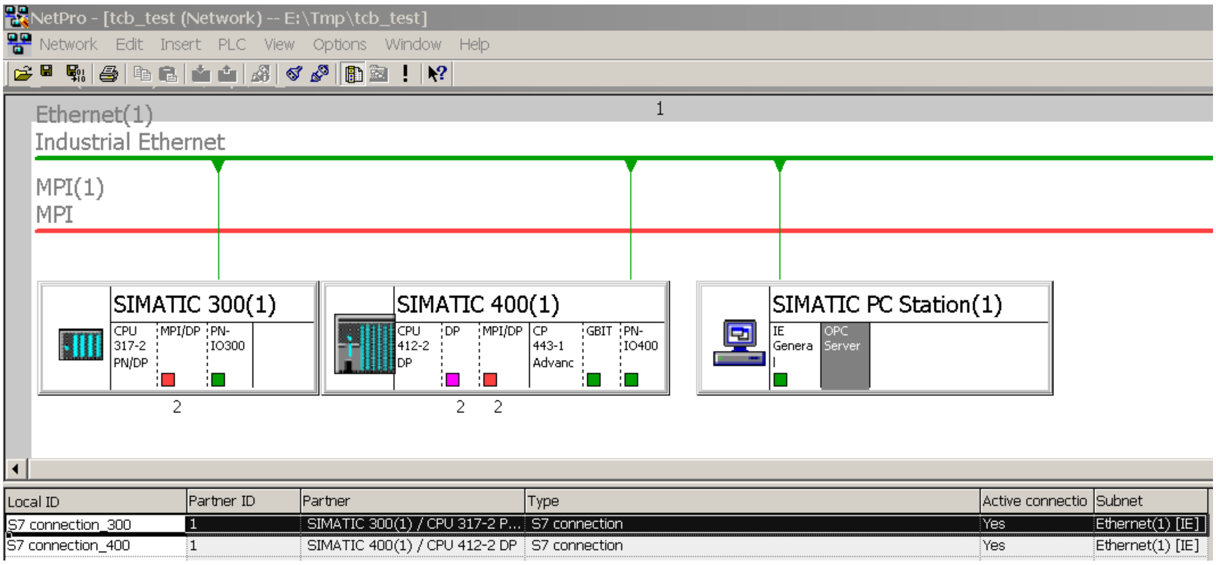


Figure 1: Topology of the test configuration

We communicate using Siemens S7 protocol over the wired Industrial Ethernet. For OPC communication it is fundamental to set the configuration properly. We have to prepare correct hardware configuration for both of PLCs and also configure OPC connections and download or import them into Simatic NET OPC server [2]. We will use one connection for each PLC. Figure 2 shows connections configuration in NetPro tool with connection names 'S7_connection_300' and 'S7_connection_400' for respective PLCs.



Local ID	Partner ID	Partner	Type	Active connectio	Subnet
S7 connection_300	1	SIMATIC 300(1) / CPU 317-2 P...	S7 connection	Yes	Ethernet(1) [IE]
S7 connection_400	1	SIMATIC 400(1) / CPU 412-2 DP	S7 connection	Yes	Ethernet(1) [IE]

Figure 2: Configuration of connections in NetPro

2 Description of the testing preparation and methods

We would like to measure the transfer time of big set of data. The way to hold larger data sets in the PLCs is the use of Data Blocks, e.g. DBs. The maximum size of DB for Simatic S7 PLCs is 64kByte. We would like to test data with largest elementary data types first. These are DINT and REAL, both represented by 32 bits in memory. DINT meaning Double Integer and REAL representing floating point number following IEEE 754 standard. We prepare a DB, named DB2, with array of 16384 DINT (or REAL) variables taking total of 64 Kbyte of memory. The code for DB2 definition in STL language follows:

```
DATA_BLOCK DB2
STRUCT
  pole: ARRAY[1..16383] of DINT;
END_STRUCT;
BEGIN
END_DATA_BLOCK
```

We would like to measure time of data transfer not only in Matlab using tic/toc commands but also in the PLC. Assume that we transfer known dataset from the Matlab to the PLC. It is a vector with ascending numbers [1:16383]. Then we can start the time measurement when element pole[1] changes its value to 1 and take the measured transfer time when pole[16383] changes to 16383. We measure the time using system function SFC64 – TIME_TCK, which reads system time with the resolution accuracy of 1 millisecond. We need to compute the time between 2 events, therefore a computation is needed. The program which is realizing this computation is out of scope of this article and an example of such a code can be found at [3]. We call the time measurement program from organization block OB1. There are also other possibilities to accomplish this task. One of them would be to use periodic organization block, e.g. OB35 with an execution period of 1 millisecond. We also would like to test if there is a difference between accesses to different data types e.g. DINT vs. REAL, which have the same data size only with different representation. Therefore we use another DB3 with array of 16383 REAL values.

On the Matlab side we need to construct an OPC Data Access object. Then there is needed to create at least one group of items and at least one item, which is an object connecting to the real OPC address pointing to a specific address in memory of PLC. After that we can connect to the OPC server and read/write values of the items/groups. The adding of items can also be done by OPC DA object connected to the OPC server but our testing showed that in this case the adding of items takes much more time. It is probably caused by the fact, that each of the items is registered on the OPC server at the time of adding. An extensive description of possibilities of OPC toolbox for data access can be found at [1]. This process can be done from Matlab command line, from a script, Simulink scheme or can be simplified using command opctool, which opens user interface to the OPC toolbox.

Test No. 1:

For the first test we used a direct definition of an array item, which is supported by OPC standard. This should be the fastest method to transfer a data vector. We have used the following code in Matlab:

```
da = opcda('localhost', 'OPC.SimaticNET') % create OPC DA object
grp300 = addgroup(da, 'TestGroup300') % create group for S7-300
grp400 = addgroup(da, 'TestGroup400') % create group for S7-400
tic %start time measurement
additem(grp300, 'S7:[S7 connection_300]DB2,D0,16383') % add item for S7-300
additem(grp400, 'S7:[S7 connection_400]DB2,D0,16383') % add item for S7-400
toc % stop time measurement
connect(da) % connect OPC DA
C=[1:1000]; %define test vector
beep
```

This code constructs OPC DA object and adds two groups, one for each PLC. Then it adds one item with a vector of 16383 DINT (or for the REAL test: 'S7:[S7 connection_400]DB3,REAL0,16383' with REAL values). We also prepare a test vector C, which will be written to the PLCs. After successful connection to the OPC server we wait for the items to connect and test synchronous write using the following code:

```
tic
write(grp300.item(1),C) %write value to S7-300
toc
tic
write(grp400.item(1),C) %write value to S7-400
toc
```

We made 10 measurements. The Table 1 shows results of average, minimum and maximum times measured in Matlab and also in the PLC with the above mentioned method.

Table 1: RESULTS OF TEST 1 – WRITE OF 16383 VALUES USING VECTOR ITEM

Data type / PLC	Matlab time [s]			PLC time [s]		
	Average	Min	Max	Average	Min	Max
DINT S7-300	**7,51**	6,37	8,68	**4,46**	4,38	4,65
DINT S7-400	**3,54**	2,98	3,91	**1,39**	1,37	1,4
REAL S7-300	**8,09**	7,2	8,59	**4,56**	4,5	4,68
REAL S7-400	**2,06**	1,72	2,59	**1,39**	1,38	1,41

Table 1 shows, that there are no significant differences between writes of different data types with the same size (DINT vs. REAL). On the other side there can be seen significant difference in the PLC times according to different PLC types. S7-400 seems to be more than twice as fast as S7-300.

We have also tested asynchronous writes using writeasync, but with very similar results on the PLC side. Of course, on the side of Matlab the times were much shorter. Using asynchronous write the script does not wait for the acknowledgement of correct write from the OPC server.

Our first idea was to continue with the next tests using the same vector size of 16k with 4-byte variables. The next test consisted of writing 16383 values. We wanted to make one item for each variable and write it as a group. But it turned out, that adding of 16383 items into OPC DA takes over 24hours. Therefore we decided to make the next tests using vectors of 1000 values only. Table 2 shows results of the writes using vector items of 1000 values each (please note, that the times measured by PLCs are in milliseconds).

Table 2: RESULTS OF TEST 1A – WRITE OF 1000 VALUES USING VECTOR ITEM

Data type / PLC	Matlab time [s]			PLC time [ms]		
	Average	Min	Max	Average	Min	Max
DINT S7-300	**5,08**	2,57	7,73	**256,2**	244	273
DINT S7-400	**3,95**	2,59	4,6	**75,8**	74	77
REAL S7-300	**4,87**	2,92	6,94	**274,8**	271	289
REAL S7-400	**4,46**	4,16	4,77	**75,8**	71	81

Test No. 2:

For the next test we will prepare two groups of 1000 items addressing the same arrays as in the test no. 1. The writing will be done by whole group to address multiple items [1]. The Matlab code for items preparation follows:

```
da = opcda('localhost', 'OPC.SimaticNET') % create OPC DA object
grp300 = addgroup(da, 'TestGroup300') % create group for S7-300
grp400 = addgroup(da, 'TestGroup400') % create group for S7-400
tic %start time measurement
for i=1:1000 % main cycle
   additem(grp300, sprintf('S7:[S7 connection_300]DB3,REAL%d',(i-1)*4)); % add item
   additem(grp400, sprintf('S7:[S7 connection_400]DB3,REAL%d',(i-1)*4)); % add item
   C{i} = i; % prepare data tu write
   if (mod(i,100))==0 i % show cycle progress
   end % if
end % main cycle
toc % stop time measurement
connect(da) % connect OPC DA
beep
```

The test vector C must be a cell vector in Matlab [1], constructed a bit different than the one used in the test no. 1. We also use a progress indicator dump after each 100 of added items.

The code for writing the item groups is the same as in the previous test with the exception that we do not write items, but only groups:

```
tic
write(grp300,C)
toc
tic
write(grp400,C)
toc
```

As mentioned above, the adding of large amount of items takes a very long time even with OPC DA object disconnected from the OPC server. In our test configuration the adding of 2x100 items (100 for each group) took approximately 30 seconds, but adding of 2x1000 items took about 300 seconds. The measured times for the test no. 2 are shown in Table 3.

Table 3: Results of test 2 – write of 1000 values using group write

Data type / PLC	Matlab time [s]			PLC time [ms]		
	Average	Min	Max	Average	Min	Max
DINT S7-300	**7,54**	6,03	8,61	**222,25**	124	263
DINT S7-400	**5,46**	4,02	6,31	**76,25**	43	90
REAL S7-300	**7,52**	6,1	8,64	**220,25**	134	269
REAL S7-400	**5,38**	4,1	6,89	**73,5**	53	92

3 Conclusions and future work

Following the measured values it can be said, that by number of 1000 variables transferred by the synchronous OPC writing there is not such a significant difference in measured transfer times. On the S7-300 CPU we have a difference of about 15-20%, with the faster S7-400 CPU we get even lower difference. We assume that by higher number of items this difference will be much higher. Our work shows, that specific data types by the same data size do not influence the transfer rates drastically. There can also be seen a significant difference between the two types of PLCs. Tests demonstrate that OPC writing performance is depending not only on the performance of the OPC server and the PC, but also on the actual PLC or endpoint device.

By the tests we used only default values of OPC communication properties, we did not experiment with values of update rate or other critical parameters. The ideas for future experiments include also tuning of update rate.

There is a wide range of practical application of our tests. The OPC transfers can be widely used by testing simulation results using real technology or by transferring of supervised setpoints or other values. The other option is data acquisition, which can be done on very fast processes by the PLC and then the measured data can be transferred to the PC using OPC channel for post processing. We have used transfers of large data sets representing speed setpoints for model of 2-axis cutting machine, which were computed using Matlab and then transferred to the controlling PLCs. More extensive research of possibilities and properties of OPC data transfer could therefore bring useful results.

Acknowledgement

This paper was supported by the Slovak Scientific Grant Agency VEGA, Grant no. 1/2256/12.

References

[1] The MathWorks, Inc. *OPC Toolbox (online documentation)*. Available on http://www.mathworks.com/help/opc/, [10-2014].

[2] Siemens AG. *OPC Simatic Net Manulas*. Available on http://support.automation.siemens.com, [10-2014].

[3] Siemens AG. *Online product support – How can you measure the time?*. Available on http://support.automation.siemens.com, [10-2014].

Mrafko Leo MSc. PhD.
leo.mrafko@stuba.sk

Kajan Martin MSc.
martin.kajan@stuba.sk

Viszus Eugen MSc.
eugen.viszus@stuba.sk

Vodička Vladislav MSc.
vladislav.vodicka@stuba.sk

All authors are members of
Institute of Robotics and Cybernetics
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology in Bratislava