Interconnecting Matlab with TwinCAT

ĽUDOVÍT FARKAS, ĽUBOSLAV JANÍČEK, JÁN MURGAŠ, JURAJ HNÁT
Institute of Control and Industrial Informatics
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology
Ilkovičova 3, 812 19 Bratislava
SLOVAK REPUBLIC
ludovit.farkas@stuba.sk, luboslavjanicek@gmail.com, jan.murgas@stuba.sk, juraj.hnat@stuba.sk

Abstract: - The use of networked control systems is very modern in industrial control. The evolution of computer networks, the increase of transfer speed and noise resistance caused the installation of Ethernet technologies in industry more frequent [1]. The most frequently used standard for interconnecting Matlab with industrial devices is the OPC standard. Our goal was to design a Matlab toolbox for communication with TwinCAT without the use of OPC.

Key-Words: - EtherCAT, TwinCAT, Matlab, Simulink, Networked Control Systems

1 Introduction

Nowadays the previous signal wiring in industrial applications is replaced by modern industrial fieldbuses. Their development has been highly influenced by the improvement of data transfers in computer networks. Some of the technologies from the computer networks found their way to industrial fieldbuses. The common reason for the development of the technologies is saving money by reduction of the wiring, simplification of the installation and increasing the reliability.

The most common, most reliable and most secure way of connecting nodes communicating over an industrial network is by cables. There are several technologies for use in industrial applications. Each of them is supported by a number of device manufacturers. As we have presented, the benefits of the networks are mainly the cost effectiveness and reliability. Each technology respects these needs. The main differences are in the transfer rates and the communication protocols. [2] It is modern to implement the technologies of wired Ethernet. One of the technologies based on Ethernet is EtherCAT.

The aim of this paper is to present our way of interconnecting Matlab with the devices connected to the EtherCAT fieldbus with a TwinCAT master. This interconnection can be convenient because of the computing and simulation capabilities of Matlab/Simulink. It can simplify design and testing of controllers, visualization, gathering of statistical data, etc.

In the paper we briefly explain the EtherCAT fieldbus and the TwinCAT softvare, then we explain the programming of MEX-files in Matlab and S-files in Simulink and we make a final presentation of our improved communication toolbox. Its first version was introduced in [3].

2 EtherCAT

The EtherCAT fieldbus has been developed by the Beckhoff automation company and the EtherCAT Technology Group (ETG). EtherCAT sets new standards for real-time performance and topology flexibility, whilst meeting or undercutting fieldbus cost levels [4]. It is also an international standard (IEC, ISO and SEMI).

EtherCAT is an open, Ethernet based technology and it implements Master – Slave communication. It uses a different way of handling the Ethernet frame as the Local Area Network (LAN). The frame is processed “on the fly”. Delay of only a few nanoseconds occurs in the fieldbus due to this way of handling the frames and the fieldbus observes hard real - time. In LAN the delay is longer and it cannot be guaranteed that the communication will observe hard real - time.

A common characteristic of the two technologies is the use of the same network cables and some other network devices, e.g. network cards. The use of these components allows easy and relatively cheap installation and the ability of further trouble free expansion of the fieldbus.

A Master device on the fieldbus is an industrial PC, on which a software PLC is installed. The Master device manages the whole control process. Slave devices are distant inputs, outputs, communication modules or servo drives connected with Ethernet cable. The connection with higher control levels is handled by the Master.

The EtherCAT fieldbus supports almost every network topology. The bus, star or tree topology can be used. It is useful to combine some bus sections with nodes. No additional switches are needed because the needed interfaces are integrated in the bus couplers. The use of switches is possible. There is a sample EtherCAT topology in Figure 1.

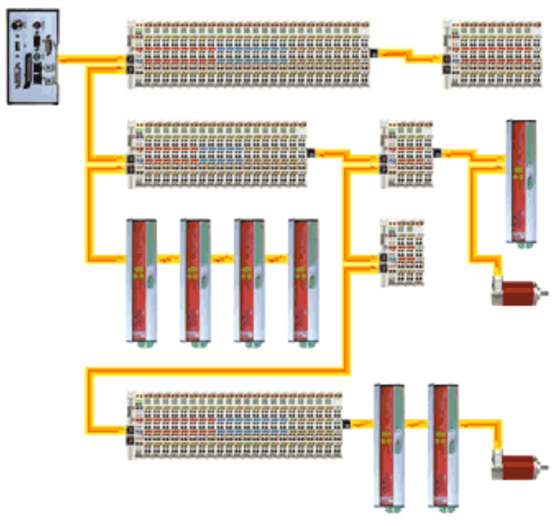
Fig. 1: Example of EtherCAT topology [5]

3 TwinCAT

TwinCAT is a real-time extension of the Microsoft Windows (XP, Vista, 7) operating system. It turns every compatible PC to a controller working in real-time. Actually it is a software PLC and it is the master software for EtherCAT. We are able to replace all common PLC systems with TwinCAT and an industrial PC. In agreement with [6], this is achieved by:

- Support of compatible PC hardware and devices
- Built in IEC 61131-3 compatible software PLC, software NC and software CNC compatible with Windows NT/2000/XP/Vista, 7, NT/XP Embedded, CE
- Programming environment and run-time system together on one PC or apart
- Connectivity to all common industrial fieldbuses
- Data communication with user interfaces and other programs by means of open standards (OPC, OPX, DLL)

The TwinCAT system consists of a run-time system (executing environment), which executes the control algorithms in real-time and of a development environment for programming or diagnostics. The run-time system is called System Manager and the development environment is called PLC Control.

The basis of TwinCAT is an accurate time base, so the PLC programs behave strictly deterministic, independently on other tasks of the processor.

Particular parts of the TwinCAT system are in [4] described as:

- **TwinCAT I/O** – part that handles the input/output interfaces of the computer
- **TwinCAT PLC** – the software PLC part of the TwinCAT system. It enables the work of four virtual PLC processors. The PLC program can be written in each language defined in the IEC 61131-3 standard
- **Operating interfaces** – for each of above-mentioned parts an operating interface is available. By these interfaces the PLC can be programmed or the whole hardware system can be configured.
- **Additional libraries and software** – the TwinCAT system can be extended with some ready made blocks, e.g. the controller blocks form the Controller Toolbox, it can be extended by additional communication capabilities (OPC server, FTP server) or some diagnostic tools.

4 Interconection with Windows applications

Because the TwinCAT system is integrated into the Windows operating system, it can use the PC resources (hard disk, network, graphics, etc.) with the methods and through the interfaces of the operating system. At the same time, the real time software has to perform certain tasks [7]:

1. Synchronisation with the operating system,
2. Adaption of data representation (data alignment),
3. Guaranteeing data consistency in the event of access.

A data interface must above all

4. Fulfill the requirements of automation,
5. Ensure full integration into the operating system.

Because of the full integration, the compatibility between TwinCAT and Windows programs is guaranteed and the use of standards is allowed. In order to allow participation in ADS communication (as an ADS client or, possibly as an ADS server) the following software objects are made available [8]:

- ADS-OCX - (ActiveX-Control) for use under e.g. Visual Basic, Visual C++, Delphi, etc.
- ADS-DLL - for use under e.g. Visual C++, etc.
- ADS-Script-DLL - for use under e.g. VBScript, JScript, etc.
- "PlcSystem.lib" - PLC library ADS services from other ADS devices can be used from the TwinCAT PLC (e.g. exchange of data with other PLC runtime systems, or calls to some of the NT functionality of the Windows NT operating systems, such as, for instance, output of message boxes).

As TwinCAT uses the message router and the message system, Windows applications can not only operate with local servers, but also with remote servers on other PCs. If we want our own program to communicate with the TwinCAT system, we have to use a specific interface made for specific programming language (C, C++, Java …). The interface is the ADS-DLL (Dynamic Link Library). This interface offers also access to the methods of the PLC (start, stop, program loading), not only to the variables.

A very common way of communication between industrial devices is the OPC standard. The OPC specifications are definitions of common interfaces to allow applications, OPC server, OPC clients and devices to exchange data, events and information [9].

In the TwinCAT system, this standard is also implemented over the ADS interface [10]. It means the data is flowing through the message router. We can say the message router handles the entire communication of the TwinCAT system.

5 Interconection with Matlab and Simulink

Matlab is a very powerful tool for technical computing and simulation. It is a time saving tool for high performance computations because it is optimized for this kinds of tasks [11].

By default Matlab allows communication with automation devices by the OPC communication standard. This is handled by the OPC Toolbox, which enables this standard also in Simulink by a library of blocks. In Matlab it is represented by a set of commands. This type of interconnection is suitable for some visualization, diagnostic or statistical purposes.

For this communication an OPC server is needed. Because the OPC server is vendor specific, you need to buy a TwinCAT OPC server licence. The TwinCAT OPC server supports OPC Data Access and Alarms & Events standards, while Matlab supports only Data Access. It allows access to TwinCAT variables by address, by name or by generic ADS Index Group and Index Offset. Today the OPC server only imports global variables and does not import structures.

The main disadvantages of use of the TwinCAT OPC server to communicate with Matlab are the need to buy the licence and the fact, that the server imports only global variables. It is also not possible to access the methods and other functions of the PLC.

To solve this problem, we had to choose another type of data exchange. Previously we mentioned the ADS interface of the TwinCAT Message Router. This interface is responsible for data exchange between the TwinCAT system and other Windows applications. We had to choose one of the ADS software objects capable to provide the data for Matlab. So we analyzed the capabilities of Matlab to write new functions and confronted it with the available ADS software objects.

A conjunction of these thoughts was to use the ADS-DLL library and to write some own MEX-files. MEX stands for MATLAB Executable. MEX-files are dynamically linked subroutines produced from C, C++ or Fortran source code that, when compiled, can be run from within MATLAB in the same way as MATLAB M-files or built-in functions. The external interface functions provide functionality to transfer data between MEX-files and MATLAB, and the ability to call MATLAB functions from C, C++ or Fortran code [12]. The ADS-DLL function can be used with the C or C++ programming languages so this integration seems to be the right way to exchange data between Matlab and TwinCAT.

We decided to make a library of functions able to handle TwinCAT data and we named it the ADS toolbox and we extended the toolbox with some Simulink blocks.

5.1 MEX-files

Usually the main reasons to write a MEX-file are [12]:

1. The ability to call large existing C or FORTRAN routines directly from MATLAB without having to rewrite them as M-files.
2. Speed; you can rewrite bottleneck computations (like for-loops) as a MEX-file for efficiency.

In our case we use MEX-files to use the possibilities of the ADS-DLL library. We know this library contains functions for accessing the variables and the functions of the TwinCAT system. It can be used with the C++ programming language, so we have written the MEX-files in that language.

We had to write the MEX-files in the way, that all of them must include four things:

1. #include mex.h – necessary to use the mx* and mex* routines
2. mexFunction gateway – the entry point for Matlab
3. the mxArray – structure containing Matlab data
4. API functions – mx* functions used to access data inside of mxArrays

We had to include also the TcAdsDef.h and TcAdsAPI.h header files provided by Beckhoff. These headers are needed to use the functionality of the TcAdsDll (ADS-DLL). With the TcAdsDll we are able to communicate with local TwinCAT systems via the Message Router or with remote TwinCAT systems via TCP/IP. The TcAdsDll provides functions to open or close the ADS port, to get the address of the device, to read and write process variables.

We are able to read and write variables not only by their address, but also by their name and against the OPC standard we can read all of the variables, not only the global variables.

Our previous version of the ADS toolbox, presented in [3], was able to read the data only in their correct C++ data types. The new improved version, introduced in [13], is rewritten and simplified for the user.

Now the user needs not to read or write the variable by its specific data type. The variable name is enough for reading or writing of the value. The port must of course be opened before the communication, and closed at the end.

There are some other additional functions written besides the opening, closing, reading and writing functions.

The MEX-files of the ADS toolbox are:

mOpenPort – opens the communication port

mClosePort – closes the communication port

mReadAllVariable – displays all variables set up in the TwinCAT system and displays their properties in a table

mReadGlobalVariables – displays the name and the properties of the global TwinCAT variables

mReadParametersOfVariable – this is a supporting function and it returns the properties of a specific variable

mReadValueByName – returns the value of the variable by its name

mReadVariableByName – returns the value and the properties of more variables

mSetValueByName – sets the value of the variable by its name

5.2 S-functions

S-functions (system-functions) provide a powerful mechanism for extending the capabilities of the Simulink environment. An S-function is a computer language description of a Simulink block written in MATLAB®, C, C++, Ada, or Fortran. C, C++, Ada, and Fortran S-functions are compiled as MEX-files using the mex utility. As with other MEX-files, S-functions are dynamically linked subroutines that the MATLAB interpreter can automatically load and execute [14].

We wanted that the users could handle TwinCAT data in Simulink simulations too. For this purpose we had to write a new library of blocks that could enable this task. We made the new library part of the ADS toolbox, so the functions for Matlab and Simulink are together. We had to use a similar principle as by the MEX-files to create new Simulink blocks. The C++ S-functions are applicable for this purpose.

The C++ S-functions are S-functions written in C++ language as the MEX-files. This allows the use of the TcAdsDll library and its functions.

We have followed the general format of the S-function writing, but we included the routines from the C++ MEX-files previously written and presented. It means we achieved a similar program structure in the whole library.

The ADS toolbox consists of 3 Simulink blocks (Figure 2):

ADS – opens the communication port at the start of the simulation and closes the port at the end of the communication

sReadValueByName – reads the value of the variable by its name, which is the parameter of the block

sSetValueByName – sets the value of the variable by its name, which is the parameter of the block.

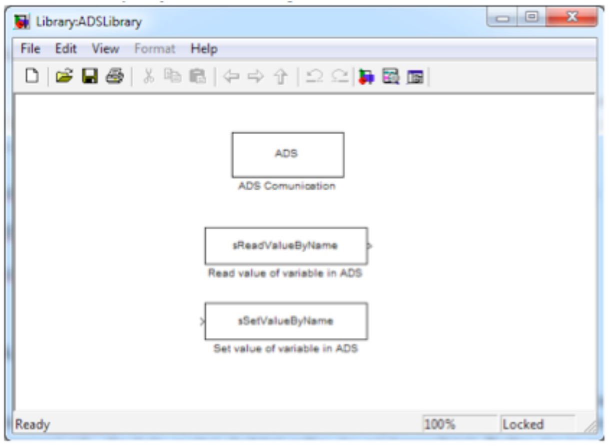
Library:ADSLibrary
File Edit View Format Help
ADS
ADS Comunication
sReadValueByName
Read value of variable in ADS
sSetValueByName
Set value of variable in ADS
Ready
100%
Locked

Fig. 2: ADS library

6 Use of the ADS Toolbox

The function *mReadParametersOfVariable* is callable at any moment, but for the use of all other funcions (*mReadAllVariables, mReadGlobalVariables, mReadValueByName, mReadVariableByName, mSetValueByName*) the *mOpenPort* and the *mClosePort* functions are needed.

The blocks from the ADS library are used in the Simulink schemes. For the real-time run of the simulation a synchronization block is needed, which slows down the simulation. Our Simulink blocks have been optimized and tested to allow accurate timing with a sample time of 0.01 s.

A sample simulation scheme is in Figure 3, where the blocks are used for plant parameters identification.

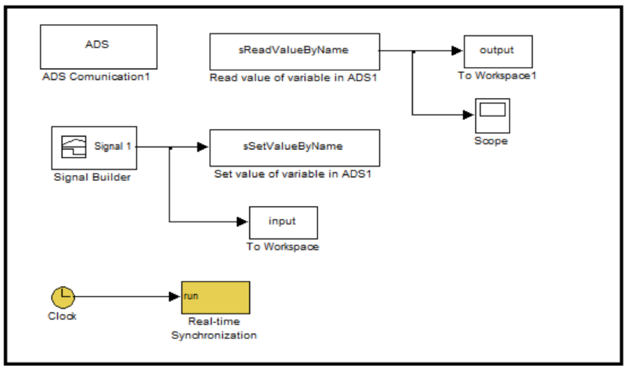

Fig. 3: Sample Simulink scheme

7 Conclusion

Our goal was to improve the previously designed Matlab toolbox to make possible the data exchange between Matlab and the EtherCAT fieldbus. We have found out that through the OPC communication standard we are not able to read all of the TwinCAT variables. We can read only the global ones. If we use the TcAdsDll library, we can read and write all of the variables and also handle some other functions of the PLC.

We have analysed the possibilities of writing our own functions and Simulink blocks using a dll library. The first version of the toolbox was introduced in [3], the new improved version was introduced in [13]. We developed a good communication tool for reading and writing variables from TwinCAT PLC. This approach is not as universal as the OPC standard, because it can communicate only with the Beckhoff ADS subsystem, but with the use of Beckhoff hardware, the need to buy the OPC server license falls out.

8 Acknowledgments

This work has received support from the Slovak Scientific Grant Agency through the grant VEGA-1/0592/10. It has been supported by the project Req-00048-001 too.

References:

[1] M. Foltin, Sieťové riadenie procesov – formulácia a trendy, *Elosys '07,* Trenčín 2007.

[2] Ľ. Farkas, J. Hnát, Simulation of Networked Control Systems using TrueTime, *Technical Computing Prague '09*, Prague, 2009

[3] Ľ. Farkas, M. Blaho, J. Hnát, Industrial Communication Between Matlab and the EtherCAT Fieldbus, *Technical Computing Prague '08,* Prague 2008

[4] *EtherCAT — Ethernet Control Automation Technology* [online], http://www.ethercat.org/, EtherCAT Technology Group, 2011.

[5] *EtherCAT - Principle of operation* [online], http://www.beckhoff.com/, Beckhoff Automation GmbH, 2011.

[6] Ľ. Farkas, *Diplomová práca – Priemyselný adaptívny regulátor založený na rozpoznávaní prechodného deja*, URPI, FEI STU Bratislava 2008.

[7] Beckhoff. *TwinCAT System Overview* [online], http://www.beckhoff.com/, Beckhoff Automation GmbH, 2011.

[8] Beckhoff. *TwinCAT ADS* [online], http://www.beckhoff.com/, Beckhoff Automation GmbH, 2011

[9] M. H. Schwarz, J. Boercsoek, A survey on OLE for Process Control (OPC), *7th WSEAS International Conference on Applied Computer Science*, Venice, Italy, November 21-23, 2007

[10] Beckhoff. *TwinCAT OPC server*[online], http://www.beckhoff.com/, Beckhoff Automation GmbH, 2011.

[11] M. Blaho et al., Preparing Advanced Matlab Users, *WSEAS Transactions on Advances in Engineering Education*, Issue 7, Volume 7, July 2010, ISSN: 1790-1979, pp. 234 - 243

[12] The MathWorks. *MEX-files Guide* [online], http://www.mathworks.com/support/tech-notes/1600/1605.html#intro, The Mathworks 2011.

[13] J. Janíček, Bakalárska práca – Komunikácia medzi Matlabom a PLC značky Beckhoff, URPI, FEI STU, Bratislava, 2011

[14] The MathWorks. *What Is an S-function* [online], http://www.mathworks.com/help/toolbox/simulink/sfg/f6-151.html, The MathWorks 2011