# INDUSTRIAL COMMUNICATION BETWEEN MATLAB AND THE ETHERCAT FIELDBUS

*Ľ. Farkas*, M.Blaho*, J. Hnát***

* Syprin, s.r.o. Systémy priemyselnej informatiky, Žehrianska 10, 851 07 Bratislava, Slovak Republic, www.syprin.sk

** Institute of Control and Industrial Informatics, Faculty of Electrical Engineering and Information Technology, Ilkovičova 3, 812 19 Bratislava, Slovak Republic

**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 EtherCAT fieldbus is one of the Ethernet based fieldbuses. The OPC standard is needed to interconnect it with Matlab/Simulink. Our goal was to design and make a Matlab toolbox to make the interconnection possible without the use of OPC.**

## 1 EtherCAT

In modern control a variety of companies develop their own fieldbus, or some of them join together to develop one. The Beckhoff automation company made the EtherCAT named industrial fieldbus. EtherCAT sets new standards for real-time performance and topology flexibility, whilst meeting or undercutting fieldbus cost levels [2]. It is also an international standard (IEC, ISO and SEMI).

Around this fieldbus a group of companies formed a global organization in which OEM, end users and technology providers join forces to support and promote the further technology development. The name of the organization is The EtherCAT Technology Group.

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". The Fieldbus Memory Management Modules (FMMUs) are used for this purpose. The FMMUs in each device on the bus read and shift the data to the device for processing and at the same time the frame is sent to the next device. 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 comparison, the delay in LAN occurs due to access to the medium and due to handle speed of the data. The handle speed is slower because the packet is received at first, then it is copied and handled at last. The mentioned sequence makes the longer delay and it cannot be guaranteed that the communication will observe hard real - time [3].

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. Example of some connected slave devices with a bus coupler is in the Figure 1. The connection with higher control levels is handled by the Master.

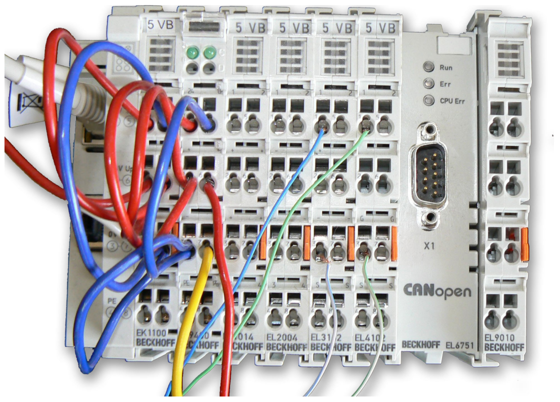
Figure 1: Interconnected slave devices. The bus coupler is on the left

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. But the use of switches is possible. There is a sample EtherCAT topology in Figure 2.

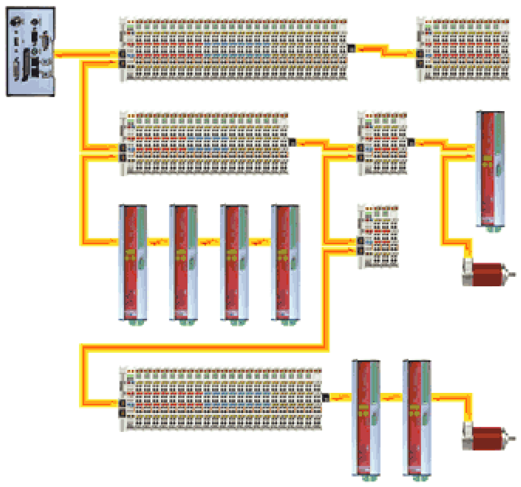
Figure 2: Example of EtherCAT topology [4]

## 2 TwinCAT

TwinCAT is a real-time extension of the Microsoft Windows (XP, Vista) operating system. It turns every compatible PC to a controller working in real-time. Actually it is a software PLC and it is the control software for EtherCAT. We are able to replace all common PLC systems with TwinCAT and an industrial PC. In agreement with [5], 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, 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 TwinCAT system is accessible from the Windows taskbar, it is shown in Figure. 3.

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 [2] 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.

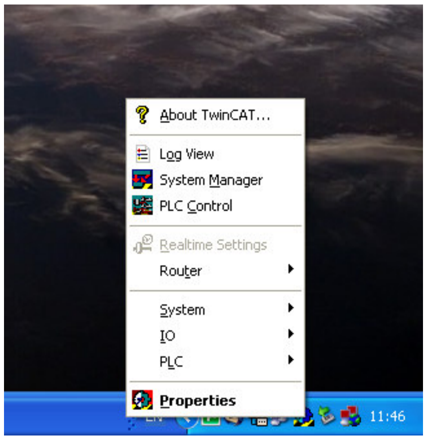

Figure 3: TwinCAT interface accessible from the taskbar

## 3 Interconnection 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 [6]:

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

A data interface must above all

- Fulfill the requirements of automation,
- 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. Of course all of the standards used for data exchange are Microsoft standards (COM, DCOM, OLE, OCX, ActiveX). These standards are implemented over the ADS (Automation Device Specification) interface of the TwinCAT message router. The message router manages and distributes all the messages in the system and over the TCP/IP connections.

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 [7]:

- 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. In the TwinCAT system, this standard is also implemented over the ADS interface. It means the data is flowing through the message router. We can say the message router handles the entire communication of the TwinCAT system. The scheme of the entire communication concept is shown in Figure 4.

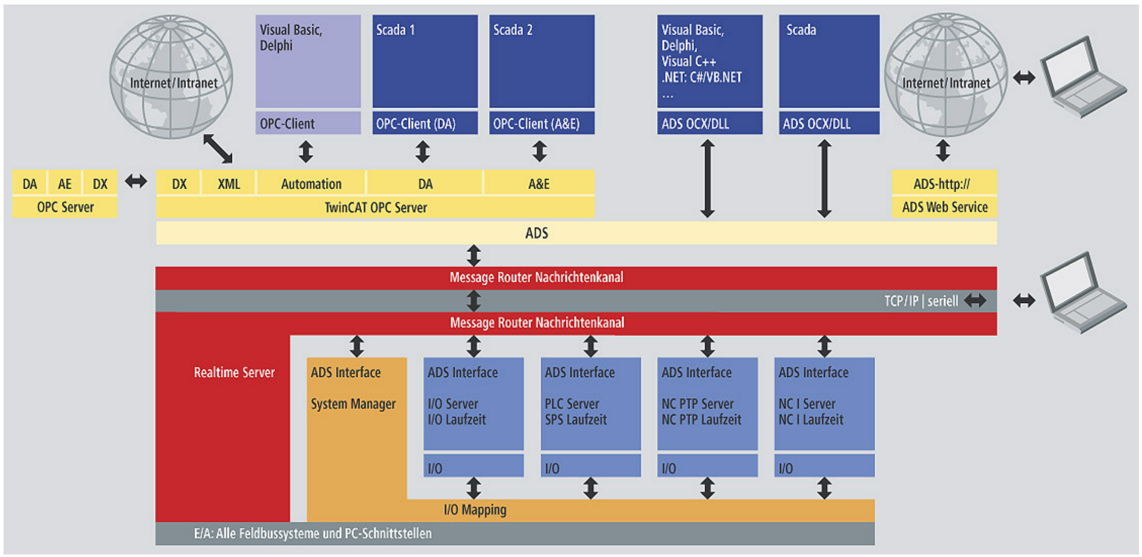

Figure 4: TwinCAT system communication concept [8]

## 4 Interconnection with Matlab/Simulink

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 is the need to buy the licence and the fact, that the sever 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 of 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 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 or Fortran code [9]. 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.

The best way to pin together a few functions is to make a library of them. So we decided to make a library of functions able to handle TwinCAT data and we named it the ADS toolbox. We wanted also that Simulink could handle data from TwinCAT, so we learned the possibilities to make new Simulink blocks through S-functions and we extended the toolbox with some Simulink blocks.

## 4.1 MEX-files

Usually the main reasons to write a MEX-file are [9]:

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.

The first step was to configure the Matlab environment, to set up the compiler. We used the Microsoft Visual C++ 2008 Express Edition compiler. So we set the proper path for it in the *mex – setup* command. After setting the right compiler, we were able to compile the MEX-files we wrote.

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. When compiling the code with the *mex* command, we have to link it with the TcAdsDll:

>> *mex sReadReal.c "TcAdsDll.lib"*

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.

As for reading the TwinCAT data types, we have to read them in the correct C data types. So we have written a special reading function for each type (readInt, readWord, readTime, etc.). An example of code for reading the variable by variable name is below:

```
  /* Fetch handle for the PLC variable */
    nErr = AdsSyncReadWriteReq(pAddr, ADSIGRP_SYM_HNDBYNAME, 0x0, 
sizeof(lHdlVar),&lHdlVar, sizeof(szVar), szVar);

    if (nErr != 0){
        mexPrintf("There was an error fetching the handle. Number of error: 
%ld \n" , nErr );
     }

    mexPrintf("Handle: %ld \n" , lHdlVar );

    /* Read value of a PLC variable (by handle) */
    nErr = AdsSyncReadReq( pAddr, ADSIGRP_SYM_VALBYHND, lHdlVar, 
sizeof(dwData), &dwData );

    if (nErr != 0){
        mexPrintf("There was an error reading the variable. Number of 
error: %ld \n" , nErr );
     } else {
        mexPrintf("Value of the variable: %ld \n" , dwData );
     }
```

## 4.2 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 [10].

We wanted that the users could handle TwinCAT data also in Simulink simulations. 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. C MEX S-functions can be created by hand-writing, by the S-function Builder or by the Legacy Code Tool. The hand-written S-functions support the widest range of features. So we have written our functions by hand.

The C MEX S-functions should be written in a general format shown below [10]:

```
#define S_FUNCTION_NAME  your_sfunction_name_here
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"

static void mdlInitializeSizes(SimStruct *S)
{
}

<additional S-function routines/code>
```

```
static void mdlTerminate(SimStruct *S)
{
}
#ifdef MATLAB_MEX_FILE    /* Is this file being compiled as a
                             MEX-file? */
#include "simulink.c"     /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"      /* Code generation registration
                             function */
#endif
```

So we followed the general format of the S-function writing, but we included the routines from the C MEX-files previously written. It means we achieved similar program structure in the whole library. We made only a few blocks in the library but it was enough for testing. The library with 3 blocks is in Figure 5.

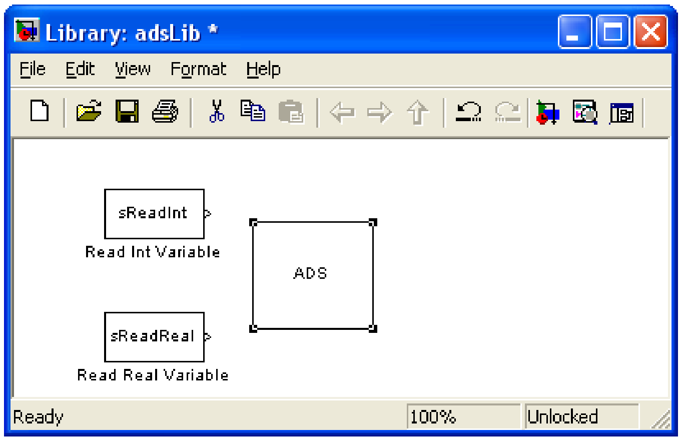

Figure 5: ADS Simulink library with 3 blocks

The ADS block must be present in every simulation scheme which uses ADS communication. It calls the openPort function and the closePort function when the simulation starts/stops. The other blocks are for variable reading. The parameters for these blocks are the names of the variables.

## 5 Conclusion

Our goal was to design and make a Matlab toolbox to make possible the data exchange between Matlab and the EtherCAT fieldbus. We showed the interconnection possibilities and discovered 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.

So we analysed the possibilities of writing own functions and Simulink blocks using a dll library. We decided to make a new toolbox by writing C MEX-files and C MEX S-functions. With this approach we achieved 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 licence falls out.

## 6 Acknowledgments

This work has received support from the Slovak Research and Development Agency. Project reference number is APVV-99-045805.

## References

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

[2] *EtherCAT — Ethernet Control Automation Technology* [online], http://www.ethercat.org/, EtherCAT Technology Group, 2008.

[3] T. Murgaš, P. Fodrek, Ľ Farkas. *Priemyselné zbernice s pripojením na Ethernet*, Elosys 2007, Trenčín 2007.

[4] *EtherCAT - Principle of operation* [online], http://www.beckhoff.com/, Beckhoff Automation GmbH, 2008.

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

[6] Beckhoff. *TwinCAT System Overview* [online], http://www.beckhoff.com/, Beckhoff Automation GmbH, 2008.

[7] Beckhoff. *TwinCAT ADS* [online], http://www.beckhoff.com/, Beckhoff Automation GmbH, 2007

[8] Beckhoff. *TwinCAT OPC server*[online], http://www.beckhoff.com/, Beckhoff Automation GmbH, 2008.

[9] The MathWorks. *MEX-files Guide* [online], http://www.mathworks.com/support/tech-notes/1600/1605.html#intro, The Mathworks 2008.

[10] The MathWorks. *What Is an S-function* [online], http://www.mathworks.com/access/helpdesk/help/toolbox/simulink/index.html?/access/helpdesk/help/toolbox/simulink/sfg/f8-48191.html&http://www.google.sk/search?hl=sk&q=mex+functions+into+simulink&btnG=H%C4%BEada%C5%A5+v+Google&meta=, The MathWorks 2008.

Ľudovít Farkas
ludovit.farkas@syprin.sk

Michal Blaho
michal.blaho@syprin.sk

Juraj Hnát
juraj.hnat@stuba.sk