Laboratory stand for multi-axis control of stepper drives via EtherCAT fieldbus

Marcin Paprocki, Krystian Erwinski
Department of Automatics and Measurement Systems, Institute of Engineering and Technology, Nicolaus Copernicus University in Torun, Grudziadzka 5/7, 87-100 Torun, Poland
E-mail: marcin.paprocki@umk.pl

Abstract.
In this paper a laboratory stand is presented that was developed for a multi-axis machine stepper drive control system. Each stepper drive utilizes an EtherCAT communication module based on a development board with software developed by the authors. The modules are used to exchange process data with a higher level controller. The EtherCAT communication standard is described including its utilization for stepper drive control. A functional description of the developed communication module is also presented. The aim of the Laboratory stand is to conduct research of cyclic communication (1 millisecond or less) and synchronization between multiple drive slave devices and a supervisory controller. The main focus of the presented results was to determine EtherCAT communication cycle jitter on the developed modules. The main aim of the research was to verify whether the data exchange via EtherCAT bus has an impact on the synchronous operation of the multi-axis system.

1. Introduction
In many areas of industrial automation, communication buses are commonly used. In recent years, networks based on the Ethernet bus have been increasingly used. A specific version of standard Ethernet is real-time Ethernet, which ensures very low delays in the network data exchange and thus obtaining transmission time determinism. There are many varieties of this Ethernet, the most important are: EtherNet/IP, PROFINET IRT, EtherCAT, POWERLINK and SERCOS III [1, 2]. One of popular standards is EtherCAT. This standard ensures high transmission speed (100 Mbps), short minimum communication cycle time (even 12 µs) and short service time for a single network node (less than 1 µs). Moreover, it implements a distributed clocks mechanism that enables synchronization of network nodes with an accuracy of 1 µs. These features are particularly desirable in systems where synchronized control of drive systems is necessary, e.g. in CNC machines [3–6] and robots [7–10]. For this reason, EtherCAT bus is one of the most widely used in distributed motion control systems.

The article presents the EtherCAT slave device communication module designed for use in stepper drives. The EtherCAT communication standard was discussed along with a description of the slave device operation using the CANopen profile. The structure and operation of the communication module are discussed. A description of the laboratory test stand for multi-axis control of stepper drives was presented along with the research. The presented EtherCAT communication module is an extension of the authors’ works in the field of drive control systems using real-time Ethernet [11–13]. The main aim of the research was to verify whether the data exchange via EtherCAT bus has an impact on the synchronous operation of the multi-axis system.
exchange via EtherCAT bus has an impact on the synchronous operation of the multi-axis system.

2. EtherCAT bus
There is one master and many slaves on the EtherCAT bus. The master device and slave devices are in the same network segment. A single network with one master device can support up to 65,535 slaves. EtherCAT uses Fast Ethernet standard (100 Mbps, UDP/IP) in full-duplex mode - one pair of wires (TX) is used to transmit data from the master to the slaves and the other (RX) to transmit in the opposite direction. A line, tree, star or ring topology can be used.

The data section in an EtherCAT frame consists of datagrams, each of which contains data for individual slaves. The data is transferred between master and slaves in the form of process data (PDO - Process Data Object). Each PDO contains information that allows to verify that the data has been correctly read and processed. The standard Fast Ethernet network controller (network card) can be used in master devices (e.g. a PC). Unlike master devices, slave devices require to use a specialized controller in their construction.

The master device is responsible for initiating the data transfer to the slave nodes. First, the master device sends one Ethernet frame that reaches all slaves. The first slave then extracts the input data from the frame and inserts the output data into the frame. The modified frame is sent to the next slave again. The end slave device sends the modified frame to the master device. From the master’s point of view, only one data frame is sent and received. The EtherCAT controller of the slave device at the time of receiving the frame only reads data from the frame and writes data to the frame. This is done "on the fly" - without the frame being analyzed and processed by the controller. The actual data processing by the protocol stack and the preparation of the feedback data takes place later. As a result, the delays introduced by the network node are minimal (several nanoseconds).

The CSMA/CD mechanism is not used in EtherCAT. The data transfer time is completely controlled by the master which should be capable of real-time deterministic operation to ensure cyclical data transfers with low latency and low jitter. In the EtherCAT bus, synchronization of network nodes is possible. For this purpose, the distributed clock mechanism (DC - Distributed Clock) is used. This mechanism allows all network nodes to be synchronized with an accuracy of less than 1 microsecond. In this mechanism, the clock of one of the slave devices in the network is a reference clock to which the clocks of the other devices are periodically synchronized. The process of clock synchronization measures network propagation delays during network initialization. Thanks to this mechanism, all devices on the EtherCAT network can be triggered simultaneously. This is important in multi-axis control, where the individual drives must move synchronously to each other.

3. EtherCAT slave communication module based on LAN9252
The EtherCAT protocol defines the physical, data link, and application layers of the OSI model. These layers are implemented differently for master and slave devices. The slave devices are equipped with a dedicated ESC (EtherCAT Slave Controller) to ensure very fast "on-the-fly" processing of data from the Ethernet frame.

In the physical layer, slave devices use the standard RJ45 (8P8C) connector and standard Ethernet PHY (physical layer) systems. For the data link layer, ESC controllers are used. Authors use the LAN9252 [14] chip from Microchip Technology, which has built-in Ethernet PHY circuits. The application layer is most often implemented by software in microprocessor systems. The authors implemented the EtherCAT application layer as dedicated software on a 32-bit microcontroller - STM32F411CE [15] from STMicroelectronics. Figure 1 shows a diagram of an EtherCAT slave communication module based on LAN9252.
The data exchange with the application layer on the microcontroller takes place using the local ESC interface, which in the EtherCAT standard is called PDI (Process Data Interface). From the point of view of the STM32F411CE microcontroller, the ESC chip is treated as an external memory that has the ability to generate interrupts. Through PDI, it is possible to read and write appropriate memory regions in ESC - data transmission from and to the microcontroller. In this work, the PDI in the SPI mode was used in LAN9252 (ESC) (Figure 1). The information exchange between devices through the SPI begins when the microcontroller sets the state "0" on the SCS_N line. Communication via PDI consists of the address sending phase and the data transfer phase to a given address in ESC. In the address phase, the master transmits the address to be accessed and the command. In the data phase, the output data sent by the slave device is presented and the input data is sent by the master device.

![Block diagram of ESC LAN9252 connection with STM32F411CE microcontroller via SPI (PDI)](image)

Figure 1. Block diagram of ESC LAN9252 connection with STM32F411CE microcontroller via SPI (PDI)

The PDI is responsible for the synchronization of work between the ESC and the microcontroller. There are three timing modes in the EtherCAT standard itself. They are FreeRun, SM-Synchron and DC-Synchron respectively. In FreeRun mode the slave device does not generate the IRQ interrupt signal via ESC. Data exchange via PDI is controlled and triggered directly by the microcontroller. In the SM-Synchronization mode, the synchronization takes place as a result of receiving an Ethernet frame by the slave device. The ESC will generate the IRQ interrupt signal to trigger the microcontroller to handle the process data. In the DC-Synchronization mode, synchronization occurs as a result of a state change on the SYNC0 line, controlled by ESC. The SYNC0 signal is generated simultaneously on all devices of the EtherCAT network. The synchronization of this signal between all slave devices was due to the operation of the distributed clocks mechanism.

There are several variants of EtherCAT application layer protocols available. The most commonly used protocol is CANOpen over EtherCAT (CoE) [16]. The class of the slave device (e.g. I/O module, electric drive, position transmitter) is defined by the so-called device profile. This profile defines an OD (Object Dictionary) structure that maps the process and configuration variables of a device. This enables the creation of a standard interface between the motion controller application and the communication layer (application layer), independent of the type and manufacturer of the master controller. In the presented communication module, the authors implemented a dedicated CoE profile designed to support a single stepper drive by each module.

4. Laboratory stand for multi-axis control of stepper drives via EtherCAT fieldbus

In order to verify the correct operation of the communication module, a laboratory stand was built with four STM32F411CE microcontroller systems, ESC LAN9252 modules and STSPIN220 stepper motor driver [17] with stepper motor (diagram of a one module in Figure 1). The EtherCAT stack application layer implemented on the STM32F411CE microcontroller uses the
Beckhoff EtherCAT Slave Stack Code Tool v.5.11 library and the authors’ own software, needed for the operation of the PDI (SPI). A PC with Windows 10 with TwinCAT v.3.1 software was used as the master controller (master device). The TwinCAT application uses a real-time micro kernel that ensures that the software runs with time determinism. In the TwinCAT application, the control software has been implemented into a laboratory stand (multi-axis control). The diagram of the laboratory stand with the four-channel oscilloscope (RIGOL MSO5074) is shown in Figure 2.

The correct operation of EtherCAT communication modules on a laboratory stand was tested during its operation (setting positions for stepper motors). All four modules exchanged information via the EtherCAT bus in the DC-Synchronization mode. The cycle time was 500 µs. The cyclicity of SYNC0 and IRQ signal generation was measured for two communication modules. The IRQ signal informs about the availability of data on the SPI bus (data from PC), and the SYNC0 signal signals the start of operation of the software controlling the STSPIN220 controller. The test results are shown in the Figure 3.
As expected, the SYNC0 signal change for both modules takes place every 500 µs, which confirms the correct operation of the communication system. The shift of the SYNC0 signal between the first module and the second module is 810 ns. Similar results are obtained for the IRQ signal for both modules. The time difference is 850 ns. The research confirms that the synchronization of the stepper drives operation at the ESC level (physical layer and data link layer) is very good (¡1 µs). Therefore, in some experimental solutions, the challenge may be to implement the software application layer of drive module. Taking into account the fact that this layer is most often implemented in a microcontroller, on which the drive control algorithm also works, it is important to maintain the timing of both tasks (communication stack and control algorithm).

At the laboratory stand, the data exchange cycle between communication modules was 500 µs. The EtherCAT standard provides for even shorter communication cycles (100 µs and less), where synchronization is likely to be kept (further research required). For such short communication cycles, communication modules can be used in servodrive solutions. This opens up new possibilities in which in the control processors of servodrives can only implement the basic functionality of such a drive (e.g. torque controller or algorithms for direct control of the power modules). The control systems themselves can then be placed in a supervisory system (e.g. at EtherCAT master system). The authors consider this an interesting direction for further research that will be continued in the future. In particular, when the communication module can be implemented in experimental drive systems for testing advanced control methods (including also for multi-axis servo systems). They can be, for example, adaptive, predictive, cross coupled, repeatable and other regulators [18, 19].

5. Conclusion
The article presents the EtherCAT bus communication module dedicated to electric drives. The module was implemented using the Microchip LAN9252 integrated circuit and the STM STM32F411CE microcontroller. The EtherCAT bus stack with the CANOpen application layer protocol has been implemented in the microcontroller. The module is designed for electric drives under development. This applies to both commercial devices as well as drives developed as part of research.

References
[1] Zurawski R 2014 Industrial communication technology handbook (CRC Press)
[2] Wilamowski B M and Irwin J D 2016 Industrial communication systems (CRC Press)
[3] Martinov G, Kozak N and Nezhmetdinov R 2017 2017 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM) (IEEE) pp 1–4
[4] Jia H, Yao P, Li B and Tian X 2017 2017 Chinese Automation Congress (CAC) (IEEE) pp 2842–2846
[5] Grigoriev S N and Martinov G M 2016 Procedia CIRP 41 858–863
[6] Li B, Lin H, Zheng L, Sun S and Yin Z 2016 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC) (IEEE) pp 795–801
[7] Liu Z, Liu N, Zhang T, Cui L and Li H 2015 2015 IEEE International Conference on Information and Automation (IEEE) pp 1708–1713
[8] Zhang G, Ni F, Li Z and Liu H 2018 2018 IEEE International Conference on Mechatronics and Automation (ICMA) (IEEE) pp 2169–2174
[9] Delgado R, Kim S Y, You B J and Choi B W 2016 2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI) (IEEE) pp 710–715
[10] Yoon D K, Kim S Y, Cho J, Lee K K and You B J 2014 2014 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI) (IEEE) pp 253–256
[11] Paprocki M, Wawrzak A, Erwinski K, Karwowski K and Klosowiak M 2017 Measurement Automation Monitoring 63
[12] Erwinski K, Paprocki M, Grzesiak L M, Karwowski K and Wawrzak A 2012 IEEE Transactions on Industrial Electronics 60 628–636
[13] Paprocki M, Wawrzak A, Erwinski K and Klosowiak M 2018 Mechanik 91 299–303
[14] Microchip Tech. 2015 2/3-Port EtherCAT Slave Controller with Integrated Ethernet PHYs
[15] STMicroelectronics 2017 Data Sheet - STM32F411CE rev. 7
[16] Chen Y, Chen H, Zhang M and Li Y 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems vol 2 (IEEE) pp 67–70
[17] STMicroelectronics 2019 Data Sheet - STSPIN220 rev. 4
[18] Szczepanski R, Tarczewski T and Grzesiak L M 2019 Applied Soft Computing 83 105644
[19] Szczepanski R, Tarczewski T and Grzesiak L 2020 Bulletin of the Polish Academy of Sciences. Technical Sciences 68