energies

MDPI

Article

Synchronization of Electrical Drives via EtherCAT Fieldbus Communication Modules

Marcin Paprocki *,† and Krystian Erwiński †

Faculty of Physics, Astronomy, and Informatics, Institute of Engineering and Technology, Nicolaus Copernicus University in Toruń, ul. Grudziądzka 5, 87-100 Toruń, Poland; erwin@umk.pl
* Correspondence: marcin.paprocki@umk.pl; Tel.: +48-56-611-2435
† These authors contributed equally to this work.

Abstract: Synchronization between devices (in particular drive systems) is paramount for multi-axis motion control systems used in Computerized Numerical Control (CNC) machines, robots, and specialized technology machines used in many areas of the manufacturing industry. EtherCAT is an Ethernet-based network that is one of the most popular industrial networks for multi-axis motion control systems. EtherCAT is standardized in the IEC 61158 and IEC 61784 standards. In the article, an EtherCAT communication network for electrical drives is presented. The article focuses on the synchronization in the EtherCAT network consisting of one master device and slave servo drive devices. Special attention is given to synchronization mechanisms in EtherCAT, such as distributed clocks in slave servo drives devices. For this purpose, a laboratory stand was built consisting of two prototype servo drive devices with BLDC motors equipped with EtherCAT communication modules. A description of the working developed EtherCAT communication modules is given. Authors in communication modules ware used an EtherCAT Slave Controller (ESC) chip (AX58100) to implement lower EtherCAT layers. EtherCAT application layer was implemented in software form on a 32-bit microcontroller, based on CANopen over EtherCAT (CoE) CAN in Automation 402 (CiA402) profile. This research's main contribution was to show the time dependencies regarding synchronization in terms of data flow in the EtherCAT communication stack in slave servo drive devices. The research results showed that the synchronous operation of drives is mainly influenced not by the mechanism of distributed clocks that ensures synchronization in the EtherCAT network but the implementation of the highest layer of the communication stack in slave servo drive devices. Experimental results are presented that prove the modules' adequacy for use in high-performance motion control systems.

Keywords: fieldbus; communication network; distributed clocks; EtherCAT; CiA402; motion control; multi-axis; synchronization

Citation: Paprocki, M.; Erwiński, K. Synchronization of Electrical Drives via EtherCAT Fieldbus Communication Modules. *Energies* **2022**, *15*, 604. https://doi.org/10.3390/en15020604

Academic Editor: Federico Barrero

Received: 27 November 2021
Accepted: 9 January 2022
Published: 15 January 2022

1. Introduction

Electrical drives are used in most areas of industry. Many applications require precise synchronization between mechanical axes controlled by electrical drives. These are CNC machines, robots, and specialized technology machines. In these applications speed and position of each axis has to be precisely matched to the speeds and positions of other axes in the machine in each time step. In the past, such synchronism was achieved by mechanical means (for example, cams), and each axis was controlled by analog signals. Nowadays, a centralized approach is common [1–6]. The central controller computes all axes' desired positions and velocities in the machine and sends them to local axis controllers (usually servo drives) in constant time intervals.

In modern industrial automation, communication fieldbuses are commonly used to connect many types of devices such as controllers, input/output modules, Human-Machine Interface (HMI) devices, sensors as well as electrical drives. An industrial fieldbus is a

network solution to provide the cyclic exchange of process data in industrial control applications. Data exchange in systems of these kinds of devices requires addressing heterogeneity and interoperability networks problems. These problems constitute an open research question in modern industry, especially for the Industry 4.0 and Industrial IoT paradigms. Several studies [7–12] focused on the role of communication networks. The quoted articles emphasized the problem of interoperability between different communication layers in the hierarchical model of the enterprise control system. In addition, it was noted that such a system in the future would not necessarily have a hierarchical form. Relatively high hopes are associated with the Open Platform Communications standard-Unified Architecture (OPC UA), which connects many networks and standard types. The OPC UA standard is currently mainly based on the Ethernet bus.

Ethernet-based communication has gained dominance in many automation solutions due to its high data transfer speed, large data payload per frame, and compatibility with existing IT solutions. Standard TCP/IP communication is not the best solution for motion control applications because of its non-deterministic characteristics. In high-performance motion control applications, multi-axis synchronization is required. Highly deterministic real-time communication fieldbuses have to be used.

T. P. Corrêa et al. [13] in their research focused mainly on reducing the communication cycle in the Ethernet network. Despite quite good delays minimization of Ethernet frames in the proposed network (of the order of 1.1 µs-from physical layer to the upper layers on user-level), the synchronization problem was partially solved (only forward synchronization). Therefore, several Industrial Ethernet standards were developed to achieve time determinism and synchronization, such as EtherNetIP [14], PROFINET IRT, EtherCAT, POWERLINK, and SERCOS III [2,3]. Xuepei Wu et al. [15] compared the three most used Industrial Ethernet standards: EtherNetIP, EtherCAT, and PROFINET IRT. The article presents a research example of DC servo control. The research results show that the best DC servo control is achieved for the EtherCAT bus (fastest relation times).

EtherCAT [16,17] is one of the most popular industrial networks in motion control solutions [18]. It retains the high data rate of other Ethernet standards (100 Mbps) but also offers very short communication cycles (as short as 12 µs) and very short handling time of each node (as short as 1 µs).

In order to enable communication according to the EtherCAT standard in newly developed or existing automation devices, a slave communication module has to be implemented. Such a module implements the EtherCAT communication protocol (fieldbus stack) without the need to increase the workload on the device's main control processor. In the case of drive systems, the specific requirements of such devices have to be taken into consideration. Also, compliance with various industrial standards has to be ensured. This is especially true in high-performance motion control applications in which precise synchronization between servo drive axes is crucial. This is important when the communication module can be used in research drive systems to verify advanced control algorithms (e.g., in multi-axis servo applications), which include: predictive, cross-coupled, repeatable, adaptive, and other regulators [19,20].

In this article, an EtherCAT module that was developed for various electrical drives is presented. EtherCAT communication standard is described. Specifically, a functional description of an electrical drive slave device communication module is given that uses the CiA402 device profile. A research stand is presented along with experimental results that focus on the synchronization performance of electrical drive axes. The main contribution of this research was to show the time dependencies regarding synchronization in terms of data flow in the EtherCAT communication stack in slave servo drive devices. The presented article is built upon the authors' earlier works in the field of using Industrial Ethernet for drive control systems [21–23].

Examples of applications using the EtherCAT bus (including issues related to synchronous data exchange) described by the researchers are presented in Section 2. The EtherCAT fieldbus is discussed in detail in Section 3. In Section 4, the authors overview

the developed EtherCAT slave module based on AX58100 and its use in a laboratory stand. Discussion of the obtained experimental results is in Section 5. Finally, the main conclusions are compiled and discussed.

2. Related works-EtherCAT synchronisation

EtherCAT bus is used in many multi-axis control applications. Therefore, to discuss the state of the art regarding synchronization in the EtherCAT bus, the authors analyzed selected articles in this field. Articles mainly related to the application were selected: multi-axis CNC applications and controlling robot drives. Finally, the authors discussed several articles directly relating to the EtherCAT DC synchronization.

For CNC applications, Martinov G.M et al. [24] described the general structure of the external CNC control system additionally equipped with a PLC program module. The developed CNC controller (AxiOMA) was equipped with a dedicated card to handle the EtherCAT stack for communication with several servo drives. This paper does not provide information on the EtherCAT bus cycle period. There were also no studies on the communication cycle and the synchronization of devices in the network.

Another example is Hao Jia et al.'s paper [25]. The authors used CODESYS software to program and CO-TRUST C37 motion controller. EtherCAT bus was used in the hardware platform. In addition to servo drives, the CNC controller supports I/O devices via the EtherCAT bus. No study of the communication cycle on the EtherCAT bus has been presented.

Li, Beibei et al. [26] presented the open CNC system architecture. The CNC controller hardware is consists of two processors architecture. EtherCAT Master stack is based on a dedicated expansion card. Research on the EtherCAT bus communication cycle has not been presented. However, the synchronization method between the Master device and the servo drive systems has been mentioned.

For robot solutions applications, Chuang, W.-L. et al. [27] designed a real-time robot control for human-robot collaboration. They used Robot Operating System (ROS) and EtherCAT bus to build this control. Robot controller acting as EtherCAT master for communication uses a dedicated PCIe expansion card based on a microcontroller with an ARM core with embedded real-time Linux software. The communication cycle in the robot application with drives was 1ms. Detailed research results on the communication cycle, including synchronization, are not presented, but it is mentioned that only Free Run mode synchronization was used.

R. Delgadol et al. [28] built a motion control system used in a mobile robot with a differential drive with the EtherCAT bus. The EtherCAT master stack is software implemented on the single board computer based on Atom D2700. The EtherCAT communication cycle in the developed system was 10 ms. The researchers also carried out a more detailed communication cycle study investigating the jitter of the communication cycle between the robot controller and the drives.

D.-K. Yoon et al. [29] built a robot arm compact embedded motor controller. A DSP-based main controller, an EtherCAT controller with a driver, and a BLDC motor driver were used to build hardware components of the controller. The software EtherCAT slave stack has been implemented on the DSP (TMS320F28335 from Texas Instruments). In the developed robot application, the drives exchange data with the robot controller device via the EtherCAT bus every 1 ms. However, the article does not present detailed studies of the communication cycle.

Zhaoming Liu et al. [30] developed the EtherCAT based robot modular joint controller that uses the EtherCAT Slave Controller ET1100 from Beckhoff. The EtherCAT slave software stack was implemented on the DSP as in the previous article. The communication cycle period was 1 ms. In the paper, the research on the communication cycle was carried out to a small extent.

Guojun Zhang et al. [31] discussed the 7-DoF Light-weight Robot with EtherCAT bus. Data transmission between joint controller and arm controller is provided via EtherCAT

bus. Field Programmable Gate Array (FPGA) is used to design the joint controller. In the device, it is possible sensor sampling and motor control. The joint controller is based on EtherCAT Slave Controller ET1100 from Beckhoff. EtherCAT slave devices communicated with the period: 1 ms, 5 ms, and 200 µs. The communication cycle research was not conducted. However, the positioning of the robot arms for different communication cycles was verified.

In the case of applications for solutions using Functional Safty, R. Delgado et al. [32] developed an embedded system (based on JECS-600ITX board) connected to EtherCAT bus. There were also drives in the network. IPCs were also used for safety-related tasks. The controller is software implemented in an embedded system based on the EtherCAT master IgH open stack. The communication cycle in the system was 1 ms. Detailed research on the communication cycle was presented.

Furthermore, the EtherCAT offers a distributed clock synchronization mechanism which is especially important in multi-axis applications like motion platforms combined with virtual reality. Lord, C.-T. et al. [33] presents a virtual reality (VR) solution based on a rotatable platform for flight simulation purposes. EtherCAT master Motion Control Card PCI-L221-P1D0 with the ability to support up to 64 axes and communication cycle time of 1ms or 0.5 ms was used.

In Table 1 a comparison of the above-mentioned scientific articles is compiled. The columns of the table include: the type of application (second column), the presented EtherCAT device: master or slave device (third column), values of the communication cycle period in EtherCAT bus, or the lack of such information (fourth column) and information about the research carried out on the synchronization aspects in EtherCAT networks. The problem of data synchronization on the EtherCAT network has been described in only a few articles.

Since 2003, when the EtherCAT standard was developed, several research teams have worked with device synchronization in this network. G.Cena et al. [34] presented a study in which they discussed the results of DC synchronization research in EtherCAT bus on a laboratory stand. The stand consisted of a master system and a few slave devices. All used devices were commercial solutions of the Beckhoff company (the authors did not analyze the device's internal structure and assumed in the experiment that they were "black box" devices). The research proved that the DC synchronization on the laboratory stand works correctly. However, some discrepancies in the results were observed with a different configuration of connections between slave devices. The authors assumed that each slave device (of the same type) should operate identically, and any DC synchronization discrepancies result only from the DC mechanism itself. However, in the study's conclusion, the authors questioned this assumption.

Another research team that worked with similar topics was the team of S.-M. Park et al., who in a series of articles [35–37] presented a new mechanism to improve DC synchronization in the EtherCAT network. The research team introduced various methods that reduced jitter between the slave and master device (as the DC master clock) in their articles. Despite demonstrating the effectiveness of each of the presented synchronization methods, the authors did not discuss the structure of used slave systems architecture. As before, they assumed that this structure does not affect the operation of the DC synchronization mechanism.

I. Kim et al. [38] developed another method to minimize jitter between the slave and master device (as the DC master clock). They present solutions with the use of the heuristic method. As before, they do not overview the slave architecture in the context of DC synchronization.

The above articles only focused on the EtherCAT master and slave cycle synchronization. Synchronizing data processing between the slaves and processing these data in the slaves' devices (between layers of the EtherCAT stack) was not discussed. For this purpose, the authors decided to build EtherCAT communication modules for an electrical device to conduct more research in the synchronization of processed data in the developed modules.

Table 1. Selected examples-applications with EtherCAT bus (CNC-multi-axis CNC applications; robots-applications for controlling robots drives; Safety-Functional Safety application; VR-virtual reality multi-axis motion solution for flight simulation purposes).

Application	Type of Application	Type of EtherCAT Device (M—Master, S—Slave)	Communication Cycle (ms) (?—Undefined)	Research on Communication Cycles (Y—Yes, N—No)
Martinov G.M et al. [24]	CNC	M	?	N
Hao Jia et al. [25]	CNC	M	?	N
S. N. Grigoriev et al. [6]	CNC	M	1–4	N
Li, Beibei et al. [26]	CNC	M	?	N
Chuang, W.-L. et al. [27]	robots	M	1	N
R. Delgadol et al. [28]	robots	M	10	N
D.-K. Yoon et al. [29]	robots	S	1	N
Zhaoming Liu et al. [30]	robots	S	1	Y
Guojun Zhang et al. [31]	robots	S	5, 1, 0.2	N
R. Delgado et al. [32]	Safety	M	1	Y
G. Peserico et al. [12]	Safety	M	1–2.5	N
Pan, C.-T. et al. [33]	VR	M	1, 0.5	N

3. EtherCAT Fieldbus

EtherCAT fieldbus uses the master-slave communication model. There is always one master device and several slave devices. The master is usually a Programmable Logic Controller (PLC) or Industrial PC (IPC) controller, and the slaves are I/O modules, sensors, or actuators. In multi-axis motion control applications, a motion controller (i.e., CNC or robot controller) is the master, and servo drives are the slaves. One master can theoretically handle up to 65535 slave devices. However, this number is limited in practice by desired minimum cycle time, required process data payload, and master controller performance. EtherCAT's physical layer is based on Fast Ethernet cabling and 8P8C (RJ45) connectors with a data rate of 100 Mbps. EtherCAT uses two out of four available twisted pairs for full-duplex transmission-one pair to send data in master-to-slaves direction and the other for data frames returning to the master. Physically the devices are usually connected in line with a dual-port switch in each slave. Due to the utilization of both signal paths in the Ethernet cable, a logical ring topology is created. EtherCAT master device can use a standard Ethernet card. The master stack can be implemented entirely in software. slaves require a specialized communication chip to implement EtherCAT's custom data link layer (OSI layer 2) called an EtherCAT Slave Controller (ESC).

Dedicated slave hardware is required because EtherCAT uses a unique frame processing method called "on the fly processing". In most industrial Ethernet variants such as Ethernet Powerlink, SERCOS III, or Profinet, the communication cycle is divided into time slots. In each time slot, the master polls each slave with a data frame and the slave responds with a return frame. This mechanism is called Time Division Multiple Access

(TDMA) and ensures time determinism which is required, especially in motion control applications. Every time the data frame is sent or received, it has to be processed by the entire master's or slaves' stack. This takes time and limits the number of slaves that can be handled in a defined cycle time. Such an approach also wastes bandwidth because each frame contains a very small amount of data, but the Ethernet frame headers, addresses, and control sum are all transmitted regardless of the data size. The EtherCAT process data for all slaves are usually encapsulated within a single Ethernet frame sent by the master at the start of each communication cycle. This frame passes through the dual-port switch of each slave. During this time, the slave hardware communication chip collects data designated for that slave from the frame and puts return data. This is done at the lowest hardware level of the stack, so the delay incurred by each slave is negligible (well below 100 nanoseconds). The actual processing of the received data and preparation of return data for the next cycle takes place in the higher layers and does not interfere with low-level processing. When the last slave receives the frame, in the next step, the frame is sent back to the master with return data updated by each slave. Such an approach allows much shorter cycle times compared to competing industrial Ethernet standards. Only the master can initialize communication, so time determinism is also achieved. This mechanism is utilized for the cyclic exchange of process data in each communication cycle, such as setpoints and actual process variable values. In addition, acyclic communication is also possible in the remaining time of the communication cycle. EtherCAT's cyclic data transfer mechanism with on-the-fly processing is presented in Figure 1.

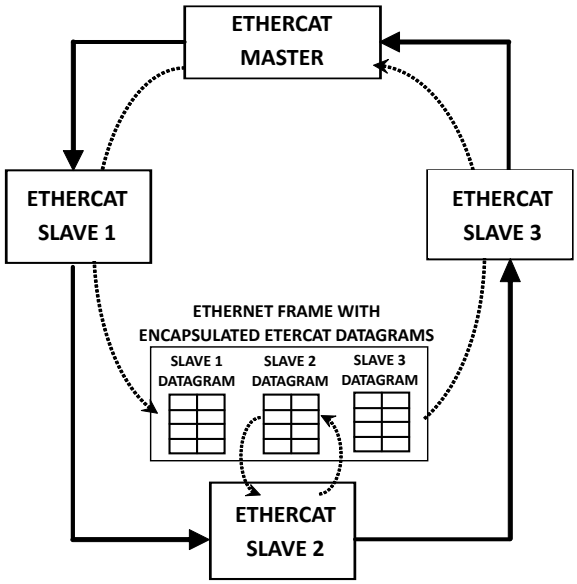

Figure 1. EtherCAT communication mechanism.

3.1. Device Synchronization in EtherCAT

EtherCAT can perform communication using one of three synchronization modes. The simplest one is the Free Run mode (no synchronization), in which the local timer of the slave runs independently of the master timer and the bus cycle. Due to the time drift of each slave and jitter of the master clock, the local timers can trigger events such as control loop computation at different times. High-performance motion control applications with cycle times well below 1 ms can lead to errors because each axis achieves the desired positions or velocities at different times. As such Free Run mode is usually used only in distributed I/O or basic motion applications.

Due to the importance of synchronizing several axes in robotics and CNC machining, the synchronization of slave devices is crucial for the EtherCAT fieldbus. Synchronization allows better control over when events such as input and output data latching are performed relative to the communication cycle. The first synchronization mode is called

SM Event Synchronization. Synchronization Manager (SM) Events are triggered when a frame passes through the device. There are two events SM2 if cyclic outputs are present and SM3 otherwise. In the case of drives, SM2 is usually used. Figure 2 presents time dependencies in this synchronization mechanism. The local slave timer is started when the frame triggers the SM2/3 event. This is usually achieved by a dedicated interrupt of the communication processor triggered by the ESC. Outputs and inputs are assumed valid at specific, well-defined points of the cycle time, which start is defined by the SM2/3 event. The received setpoint data is always synchronized with the control loop of the main drive processor. This ensures that the sent and received process data values are not lost or processed twice due to bus and processor cycle mismatch. The moment of latching input data (i.e., drive actual values) can be shifted to be as close to the next SM event as possible.

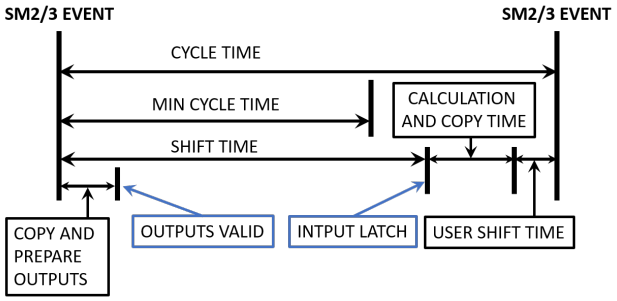

Figure 2. EtherCAT synchronization mechanisms-Sync Manager Event synchronization (SM sync) [39].

SM Event Synchronization works well if the master's clock is stable and has a very low jitter compared to the cycle time and frame propagation delays of the network are negligible [39]. Otherwise, the actual slaves' cycles can vary depending on when the frame reaches their respective ESC's. Due to the prevalence of Industrial PCs in high-performance motion control, which perform many complex control tasks, low jitter cannot always be guaranteed. Also, large networks with many different slaves and long cable propagation delays can play a significant role. This is especially true for short cycle times in the order of 100 µs. In this case, the Distributed Clock (DC) synchronization is recommended.

In DC synchronization mode, a global system time is defined (Figure 3). The first slave is assumed to hold the reference time. Usually, the slave controller is an embedded device whose clock is assumed to be more precise and stable than the master's (often PC-based). Each slave holds the copy of the system time sent by the master at bus startup. The network is then mapped by the master by sending several test frames and recording times of their arrival in the forward and return paths by each slave. This information is sent back to the master, which computes time offsets corresponding to propagation delays of each slave in the EtherCAT network. The local timers synchronized to the global timer measure the bus period. They are set to periodically trigger to interrupt an event called SYNC0. This event is offset from the start of the cycle measured by each local timer by different time offsets in each slave that compensate for master clock jitter and further propagation delays. The idea is to trigger the SYNC0 event simultaneously in each slave but only after the data frame from the master arrives at the given slave and is processed by the ESC. During the cyclic phase, the global time is periodically read from the reference slave clock and sent to other slaves to resynchronize their clocks to the global clock. This ensures compensation of clock drift. In electrical drives, SYNC0 is used similarly to the SM event to synchronize the control loop with the communication bus. Such advanced mechanisms provide excellent synchronization of drives in the EtherCAT fieldbus required by high-performance multi-axis applications where the exact following of a motion trajectory has to be ensured.

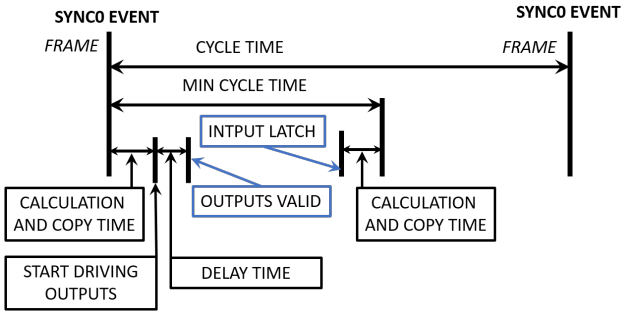

Figure 3. EtherCAT synchronization mechanisms-Distributed Clock synchronization (DC sync) [39].

3.2. CiA402 Communication Profile for Electrical Drives

To build a fully functional EtherCAT slave, the ESC chip needs an application layer (OSI layer 7) usually implemented in a communication processor. EtherCAT uses CANopen and, more specifically, CANopen over EtherCAT (CoE) [40], as the application layer, which interfaces with the actual master or slave device control program. CANopen was developed initially for CAN-based fieldbus networks. It is now also used in EtherCAT and Ethernet Powerlink fieldbuses. CANopen defines a data structure containing all data relevant to an Object Dictionary (OD) communication. Every entry in the object dictionary is given an address called an index, and sometimes a sub-address called a subindex. Each of the OD objects consists of 16-bit and data indexes. Addresses between 1000h and 1FFFh contain communication objects. Addresses between 2000h and 5999h contain manufacturer specific objects, and addresses from 6000h contain deviceprofile objects.

CANopen also defines two structures for data exchange (Figure 4). Process Data Object (PDO) is the structure containing Object Dictionary entries that are cyclically transferred as process variables. Before cyclic communication is started in the configuration phase, particular object dictionary objects are mapped to this structure. Each PDO entry has a defined offset in the dataset encapsulated in the Ethernet frame. This is important so that during the cyclic phase, the slaves' hardware knows precisely where in the frame the relevant data is located. After cyclic communication is started, PDO entries are exchanged between master and slaves in every cycle and cannot be changed without reconfiguring the communication configuration of the network. Service Data Objects (SDO) contains object dictionary entries sent and received acyclically. SDO works as a mailbox buffering received data and sending data. This communication is acyclic and is dependant on available bandwidth in the communication cycle. This communication is not deterministic and is best suited for transmitting configuration data.

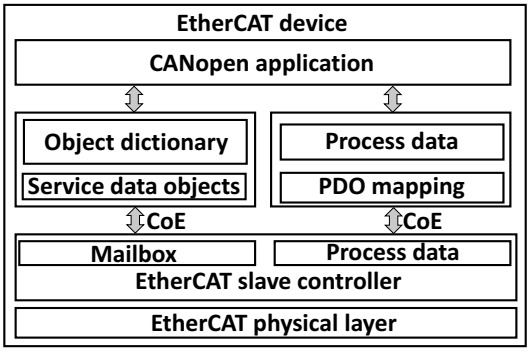

Figure 4. CANopen has two structures for data exchang-Service Data Objects and Process Data Object.

CANopen defines several device profiles which establish standard objects and behavior for similar device classes. For electrical drives, this profile is CiA402 "CANopen device profile for drives and motion control", which is defined by the IEC standard [41]. Index objects in Object Dictionary in the range 6000h to 67FEh are specified by the CiA402 profile. If a drive is compatible with CiA402, typical process values such as position, velocity, or torque setpoints and actual values will always be defined in the same objects with the same address regardless of the manufacturer. This dramatically simplifies the commissioning of different drives in a motion control system. CIA402 also defines a state machine that corresponds to the drive's actual operating state (Disabled, Faulted, Switched ON, etc.). This state machine is controlled and monitored by two mandatory PDO entries: Control Word and Status Word. The drive cannot be started until the state machine has been put into the appropriate state. Figure 5 shows the CiA402 communication profile state machine.

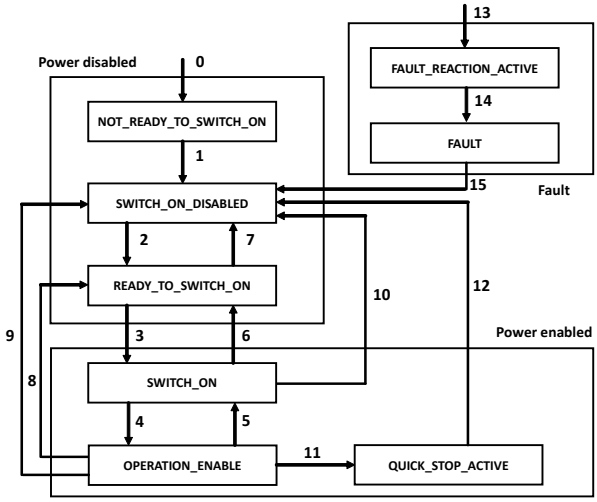

Figure 5. CiA402 communication profile state machine.

The state machine transitions depend on the 6040h index value named ControlWord. The NOT_READY_TO_SWITCH _ON state of the state machine is an internal state in which communication is enabled. The user cannot monitor this state. SWITCH_ON_DISABLED is the minimum state that the user has access to. In this state, drive initialization is complete. Drive can perform "0" and "1" transitions after auto initialization. The voltage in the drive converter system may appear in the READY_TO_SWITCH_ON state, but further operations on the drive require a change to the next state. SWITCHED_ON is the state where the drive is ready but still cannot receive motion commands. Only the OPERATION_ENABLE state allows the drive to be fully used for operation. As a result of an emergency in the drive, regardless of the current machine state, a jump to the FAULT state has proceeded.

Received PDO entries and SDO entries directly affect the servo drive's process variables such as position, velocity or torque setpoints, mode of operation. Transmitted PDO and SDO entries are directly copied from actual process variable values such as actual position, velocity, current, following error. An example of a slave's CIA402 structure in a drive's communication processor is presented in Figure 6.

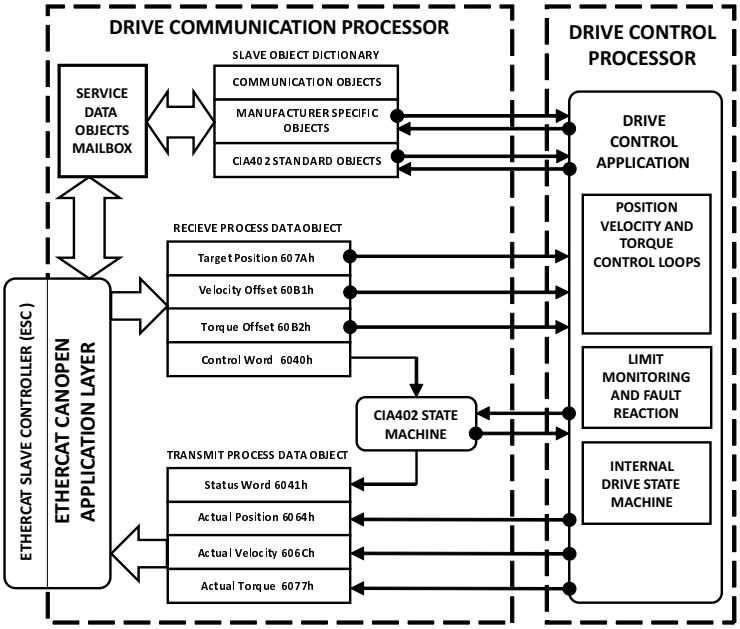

Figure 6. Example implementation of CIA402 object dictionary in drives.

4. EtherCAT Slave Module Based on AX58100

The ESC chip was used to construct the EtherCAT slave devices. The chip provides ("on-the-fly") PDO data capture from the Ethernet frame. The slave device consists of two RJ45 (8P8C) connectors with integrated transformers (Physical Layer of the EtherCAT stack). Data Link Layer (DLL) in slave device is based on the ESC AX58100 chip from ASIX Electronics Corp chip. This chip also has built-in standard Ethernet PHY. In many slave devices, the Application Layer of the EtherCAT stack is very often in a software form and is implemented in microprocessor systems. In this work, the authors implemented this Application Layer on a 32-bit STM32F303RBT6 microcontroller from STMicroelectronics in the communication module. In the presented solution, the authors also implemented the CiA402 profile of the EtherCAT stack (for electric drives control purposes). The elaborated EtherCAT slave communication module with AX58100 ESC and STM32F303RBT6 microcontroller diagram is shown in Figure 7.

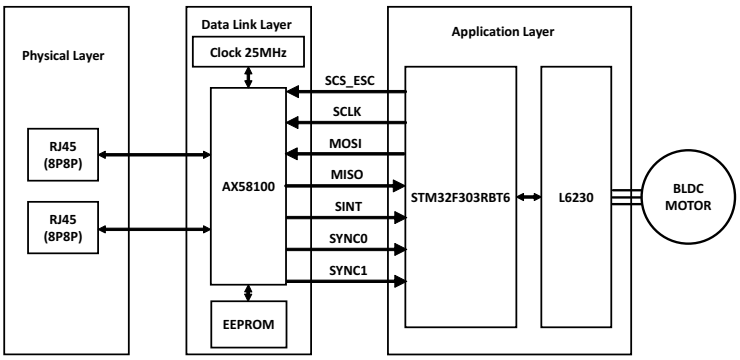

Figure 7. The EtherCAT slave communication module with AX58100 ESC and STM32F303RBT6 microcontroller diagram.

Process Data Interface (PDI) is called local ESC interface that provides data exchange with the microcontroller with AL. From the STM32F303RBT6 microcontroller perspective, access to the ESC chip is treated as access to external memory. ESC is connected with the microcontroller also using interrupts signals. The PDI interface makes it possible to read and write data from the ESC chip address space and exchange it with the microcontroller for further processing.

The AX58100 chip has a 9 KB address space. The first 1 KB (0000h-0FFFh) is dedicated to the ESC configuration registers. The process data RAM starts at the address 1000h. The RAM is read and written by the EtherCAT master or by the microcontroller via the PDI interface. ESC must be adequately initialized each time at the start because the configuration data is not saved in it permanently. Therefore, each ESC chip requires an external EEPROM to store the configuration data (I2C interface). The ESC system will not start to work until its correct configuration from EEPROM memory is read and set up correctly.

Several PDI interfaces can be used in the ESC AX58100 controller-serial SPI interface, parallel interface, or digital inputs and outputs. Registers from 0140h to 0141h are responsible for the PDI configuration in the ESC system. In the elaborate EtherCAT slave devices module, an SPI interface is used as a PDI interface (Figure 7). The SPI master is a microcontroller, and ESC is an SPI slave. When the output SCS_ESC line of the microcontroller is set to state "0", a data exchange process is started between both devices. Writing data to a specific address registers in the ESC chip is a two-phase process. First, an address data is sent, and then—data transfer phase occurs.

The PDI interface provides synchronization between the microcontroller and the ESC. As mentioned in earlier sections of this article, the EtherCAT bus supports three synchronous network data exchange modes. These modes are: FreeRun, SM, and DC. Generating SINT interrupt signal by slave device in FreeRun mode is not occur (Figure 7). However, in SM mode, an interrupt (SINT interrupt signal) is generated by ESC to handle the PDO data by the microcontroller. The source of this interrupt is the prompt of an Ethernet frame received by the slave device. There is another source of interrupt signal synchronization in DC mode-the SYNCx signals. Thanks to the distributed clock mechanism, these signals are generated simultaneously for all EtherCAT slave devices in the network.

4.1. Laboratory Stand

The EtherCAT communication modules were built for electrical drives. A laboratory stand was made with one EtherCAT master device and two Brushless DC (BLDC) servo drive system stands to verify the synchronization between communication modules. In Figure 8 laboratory stand diagram is presented. In Figure 9 laboratory stand photograph is presented.

One servo drive stand consists of STM32F303RBT6 microcontroller (NUCLEO-F303RE board), ESC AX58100 modules (AX58100-EVB-SSPDI-1 board), and L6230 three-phase brushless DC motor driver (X-NUCLEO-IHM07M1 board) with BR2804-1700 BLDC motor. On the STM32F303RBT6 microcontroller, the AL EtherCAT stack implementation uses the EtherCAT Slave Stack Code Tool v.5.11 library from Beckhoff and software needed to operate the PDI (SPI). To build a closed control loop for BLDC motor speed controls, the library MotorControl Workbench by STMicroelectronics was used. All sources were written in C language. The whole project (EtherCAT communication and cascade speed controller) was compiled using the STM32CubeIDE environment (GCC compiler, O3 optimization, and debugger flags switch off).

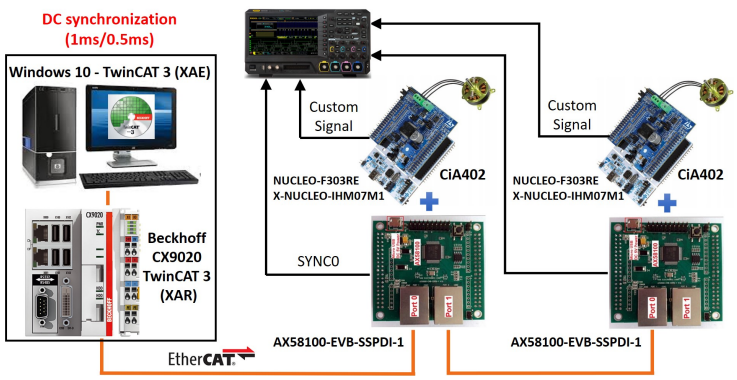

Figure 8. DLaboratory stand diagram.

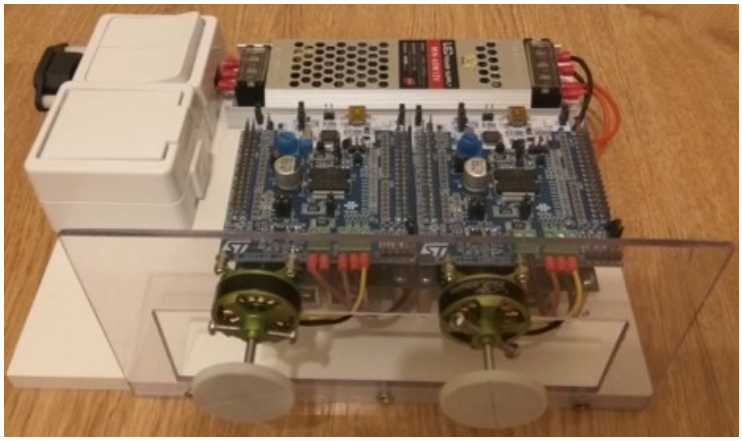

Figure 9. Photograph of the laboratory stand.

EtherCAT master device runs at Beckhoff CX9020 IPC device. On that device, a TwinCAT 3.1 eXtended Automation Runtime (XAR) application is installed. The CX9020 IPC device is connected to a standard PC with Windows 10, where the TwinCAT 3.1 eXtended Automation Engineering (XAE) application is installed. In this case, the authors built a PLC project using the TwinCAT XAE version to set the speed to the servo drives stand. The authors decided to use such an architecture of the EtherCAT master system because it ensures stable operation and minimizes the appearance of high communication jitter of EtherCAT frames generated by the EtherCAT master device.

5. Experimental Results

The experimental research focused on the study of the synchronous operation of both servo drives. Two types of tests were carried out on a laboratory stand. The first focuses on whether the data received from the EtherCAT bus is equally processed in the data link layer for both control drives (in both ESC chips). In the second experiment, the processing time measured of data transfer in the DLL to the AL direction (until processed by the regulator). All tests were carried out for two different EtherCAT communication cycles-1 ms and 500 µs (requirements of motion control-to synchronize several axes over a network [42]). Due to the easy measurement of the output signals from the ESC system and the microcontroller (SINT, SYNC0, and SYNC1), the authors decided to test only in the DC synchronization mode.

In the first experiment, the synchronous operation of both servo drives was investigated. The research focused on measuring the cyclicity of SYNC0 signal generation from two servo drives (Figure 8). The SINT signal occurs when data on the SPI bus is available (data for the SPI master-microcontroller). The SYNC0 signals indicate the moment when these data should be processed. While the SINT signals can be generated at different times for both servo drives, in the case of SYNC0 signals, they should occur simultaneously for both servo drives. The test results are presented in Table 2. The results were compiled from 1000 measurements of the SYNC0 signal (the number of measured samples is limited due to the limitations of the measurement equipment). Figure 10 shows the one oscillogram from the tests on the laboratory stand for the generation of SYNC0 signals for both servo drives (DC synchronization, EtherCAT cycle communication-500 µs).

Table 2. Time dependence measurements for SYNC0 servodrive signals (DC synchronization).

EtherCAT Cycle (µs)	Average Value Time (µs)	Maximum Value of Time Difference (µs)	Minimum Value of Time Difference (µs)	Standard Deviation Value of Time Difference (µs)
1000	805.5	831.5	730.0	4.621
500	807.2	831.5	760.0	3.725

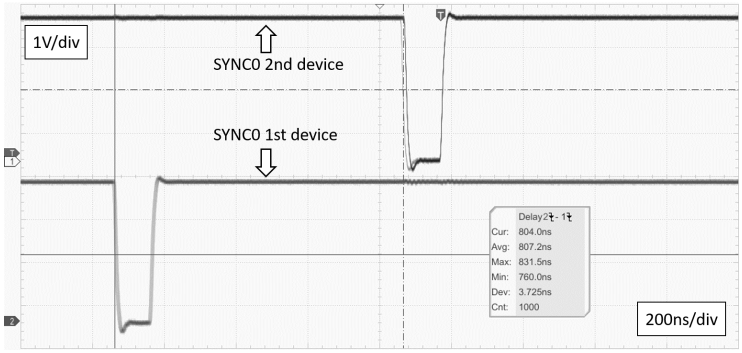

Figure 10. Time dependence measurements for SYNC0 servodrive signals (DC synchronisation, EtherCAT cycle communication-500 µs).

The SYNC0 signal occurs in both modules in every EtherCAT communication cycle (1000 µs and 500 µs). Such work confirms correct cyclic and synchronous communication of devices in the EtherCAT network. The difference time between the SYNC0 signals occurrence in both modules is in worst case 831.5 ns (in 1000 µs and 500 µs EtherCAT communication cycles) and in best case 730ns (in 1000 µs EtherCAT communication cycles). The researches prove that the servo drives synchronization at the ESC level (between the physical layer and data link layer) has a minimal delay (<1 µs) for motion control requirements [42].

In the second experiment, the time to process data in the EtherCAT application layer in each servo drive was investigated. The research focused on the measurement of duration time of a high state of "Custom Signal" (Figure 8). This signal is set (high state) when the interrupt services procedure is realized (when SYNC0 has occurred) and reset (low state) when the value "Target Velocity" is sent to the servo drive regulator. The test results are presented in the Table 3 for 1st servo drive and Table 4 for 2nd servo drive. The results were compiled from 1000 measurements of the "Custom Signal". Figure 11 shows the oscillogram from the tests on the laboratory stand for the generation of "Custom Signals" for both servo drives (DC synchronization, EtherCAT cycle communication-500 µs).

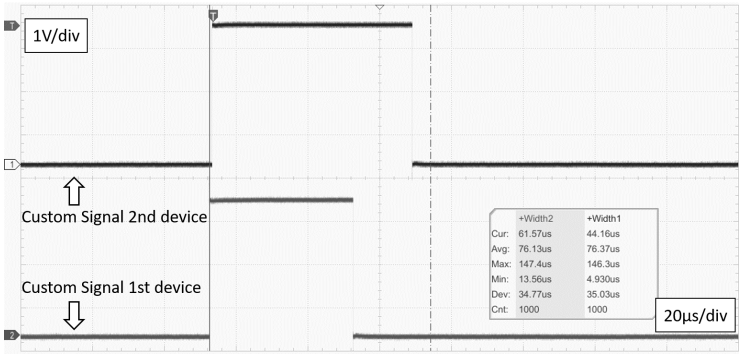

Figure 11. Maximum time to processed data in application layer in EtherCAT servodrive (DC synchronisation, EtherCAT cycle communication-500 µs).

Table 3. Maximum time to EtherCAT processed data in application layer in 1st servodrive (DC synchronization).

EtherCAT Cycle (µs)	Average Value Time (µs)	Maximum Value of Time Difference (µs)	Minimum Value of Time Difference (µs)	Standard Deviation Value of Time Difference (µs)
1000	50.49	146.0	2.93	30.30
500	76.37	146.3	4.93	35.03

Table 4. Maximum time to EtherCAT processed data in application layer in 2nd servodrive (DC synchronization).

EtherCAT Cycle (µs)	Average Value Time (µs)	Maximum Value of Time Difference (µs)	Minimum Value of Time Difference (µs)	Standard Deviation Value of Time Difference (µs)
1000	52.44	151.6	2.805	32.40
500	76.13	147.4	13.56	34.77

The duration of the "Custom Signal" (average value) takes nearly the same amount of time for both servo drives. This time (in all cases), it is dependent on EtherCAT communication cycles. For EtherCAT communication cycles of 500 µs, handling of switching "Custom Signal" output state takes average longer time than for the same operation in a cycle of 1000 µs. The previous experiment showed that data transfer from the EtherCAT ESC (physical and transport layer) has a lower average jitter than synchronous processing of switching "Custom Signal" in both microcontrollers (CoE application layer-CiA402) resulting from the second experiment. Experiment two confirms that any timing problems in EtherCAT slaves depend on the application layer implementation of the EtherCAT stack in each device. The research also confirms that the operation of the EtherCAT synchronization process (Distributed Clock synchronization) is correct and the communication cycle jitter at this level is less than 1 µs.

Therefore, when considering the build of a servo drive (especially an experimental application) equipped with an EtherCAT slave interface (with ESC chip), the concept of implementation and the related challenges should be analyzed quite strongly. In particular, the method of implementation of the application layer of the EtherCAT stack should be considered in the context of integration with control systems and data exchange with the ESC chip. In the case of implementing the AL stack in the same place (e.g., in a microcontroller), particular attention must be paid to the time dependencies of data processing, and hardware interrupts of the communication stack and control algorithms.

6. Conclusions

The authors describe the electric drives communication in the EtherCAT network in the paper. In particular, the aspect of synchronizing data transfer in the EtherCAT network. Most of the literature presents research results on synchronization between EtherCAT master and slaves. On the other hand, the article's authors focus attention on synchronization mechanisms in EtherCAT (including DC synchronization), Emphasizing slave devices-in this case, servo drives. The presented research was carried out on the laboratory stand consisting of one master device and two slave servo drives. As a result of the research, it can be emphasized that:

- EtherCAT communication modules for electrical drives were presented. The module was implemented using the ESC AX58100 chip and the STM32F303RBT6 microcontroller. The EtherCAT stack has been implemented into the module microcontroller, especially the application layer with the CANopen protocol (CiA402). The module can be used to build experimental electric drives and use with commercial devices as part of the research. The data exchange between communication modules via EtherCAT bus was carried out at the laboratory stand for 1000 µs and 500 µs cycles. Presented experimental results prove the modules' adequacy for use in high-performance motion control systems.
- It has been confirmed that the synchronization of EtherCAT slave devices is also influenced by the implementation of the EtherCAT stack (in particular, the application layer). The research results showed that the synchronous operation of drives is primarily influenced not by the distributed clock mechanism ensuring synchronization in the EtherCAT network but by implementing the highest layer of the communication stack in slave servo drive devices. In the case of servo drives, the data exchange

between the communication stack and the drive control system also affects the correct synchronization with other devices.

Although synchronization devices in short communication cycles (100 µs and less) in the EtherCAT network are possible, it requires further research. This aspect is of particular importance at servo drive solutions. High-speed synchronous data exchange opens up new possibilities for constructing new servo drive control systems. This opportunity allows limit servo drives controller to the fundamental functionality of such a drive (e.g., direct control algorithms for the power modules or torque controller). In this approach, the remaining control systems can be moved to an external supervisory system (e.g., at EtherCAT master device). This is an interesting direction for further research that the authors consider continuing in the future.

Author Contributions: Conceptualization, M.P. and K.E.; methodology, M.P.; software, M.P.; validation, M.P. and K.E.; formal analysis, M.P. and K.E.; resources, M.P.; writing—original draft preparation, M.P.; writing—review and editing, K.E.; visualization, M.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the basic research fund of the Department of Automatics and Measurement Systems, Nicolaus Copernicus University, Poland.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhou, N.; Li, D. Cyber-Physical Co-Design of Field-Level Reconfigurations in Networked Motion Controllers. *IEEE/ASME Trans. Mechatronics* **2020**, *26*, 2092–2103. [CrossRef]
2. Zurawski, R. *Industrial Communication Technology Handbook*; CRC Press: Boca Raton, FL, USA, 2015.
3. Wilamowski, B.M.; Irwin, J.D. *Industrial Communication Systems*; CRC Press: Boca Raton, FL, USA, 2011.
4. Yu, D.; Hu, Y.; Xu, X.W.; Huang, Y.; Du, S. An open CNC system based on component technology. *IEEE Trans. Autom. Sci. Eng.* **2009**, *6*, 302–310.
5. Fischer, H.; Vulliez, M.; Laguillaumie, P.; Vulliez, P.; Gazeau, J.P. RTRobMultiAxisControl: A framework for real-time multiaxis and multirobot control. *IEEE Trans. Autom. Sci. Eng.* **2019**, *16*, 1205–1217. [CrossRef]
6. Grigoriev, S.N.; Martinov, G.M. The control platform for decomposition and synthesis of specialized CNC systems. *Procedia CIRP* **2016**, *41*, 858–863. [CrossRef]
7. Scanzio, S.; Wisniewski, L.; Gaj, P. Heterogeneous and dependable networks in industry—A survey. *Comput. Ind.* **2021**, *125*, 103388. [CrossRef]
8. González, I.; Calderón, A.J.; Figueiredo, J.; Sousa, J. A literature survey on open platform communications (OPC) applied to advanced industrial environments. *Electronics* **2019**, *8*, 510. [CrossRef]
9. Zeid, A.; Sundaram, S.; Moghaddam, M.; Kamarthi, S.; Marion, T. Interoperability in smart manufacturing: Research challenges. *Machines* **2019**, *7*, 21. [CrossRef]
10. Colombo, A.W.; Karnouskos, S.; Kaynak, O.; Shi, Y.; Yin, S. Industrial cyberphysical systems: A backbone of the fourth industrial revolution. *IEEE Ind. Electron. Mag.* **2017**, *11*, 6–16. [CrossRef]
11. Wollschlaeger, M.; Sauter, T.; Jasperneite, J. The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0. *IEEE Ind. Electron. Mag.* **2017**, *11*, 17–27. [CrossRef]
12. Peserico, G.; Morato, A.; Tramarin, F.; Vitturi, S. Functional Safety Networks and Protocols in the Industrial Internet of Things Era. *Sensors* **2021**, *21*, 6073. [CrossRef]
13. P Corrêa, T.; Almeida, L. Hardware Support to Minimize the End-to-End Delay in Ethernet-Based Ring Networks. *Electronics* **2019**, *8*, 1097. [CrossRef]
14. Saez, M.; Maturana, F.P.; Barton, K.; Tilbury, D.M. Real-time manufacturing machine and system performance monitoring using internet of things. *IEEE Trans. Autom. Sci. Eng.* **2018**, *15*, 1735–1748. [CrossRef]
15. Wu, X.; Xie, L. Performance evaluation of industrial Ethernet protocols for networked control application. *Control Eng. Pract.* **2019**, *84*, 208–217. [CrossRef]
16. *Standard IEC 61158-1:2019*; Industrial Communication Networks-Fieldbus Specifications-Part 1: Overview and Guidance for the IEC 61158 and IEC 61784 Series. International Electrotechnical Commission: Geneva, Switzerland, 2019.
17. *Standard IEC 61784-5-12:2018*; Industrial Communication Networks-Profiles-Part 5-12: Installation of Fieldbuses-Installation Profiles for CPF 12. International Electrotechnical Commission: Geneva, Switzerland, 2018.

18. Stój, J. Cost-Effective Hot-Standby Redundancy With Synchronization Using EtherCAT and Real-Time Ethernet Protocols. *IEEE Trans. Autom. Sci. Eng.* **2020**, *18*, 203–2047. [CrossRef]
19. Szczepanski, R.; Tarczewski, T.; Grzesiak, L.M. Adaptive state feedback speed controller for PMSM based on Artificial Bee Colony algorithm. *Appl. Soft Comput.* **2019**, *83*, 105644. [CrossRef]
20. Szczepanski, R.; Tarczewski, T.; Grzesiak, L. PMSM drive with adaptive state feedback speed controller. *Bull. Pol. Acad. Sci. Tech. Sci.* **2020**, *68*, 1009–1017.
21. Paprocki, M.; Wawrzak, A.; Erwinski, K.; Karwowski, K.; Klosowiak, M. PC-based CNC machine control system with LinuxCNC software. *Meas. Autom. Monit.* **2017**, *63*, 15–19.
22. Erwinski, K.; Paprocki, M.; Grzesiak, L.M.; Karwowski, K.; Wawrzak, A. Application of ethernet powerlink for communication in a linux rtai open cnc system. *IEEE Trans. Ind. Electron.* **2012**, *60*, 628–636. [CrossRef]
23. Paprocki, M.; Wawrzak, A.; Erwinski, K.; Klosowiak, M. Flexible PC-based CNC machine control system. *Mechanik* **2018**, *91*, 299–303. [CrossRef]
24. Martinov, G.; Kozak, N.; Nezhmetdinov, R. Implementation of control for peripheral machine equipment based on the external soft PLC integrated with CNC. In Proceedings of the 2017 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), St. Petersburg, Russia, 16–19 May 2017; pp. 1–4.
25. Jia, H.; Yao, P.; Li, B.; Tian, X. Four axes wear-resistant coating testing system based on EtherCAT. In Proceedings of the 2017 Chinese Automation Congress (CAC), Jinan, China, 20–22 October 2017; pp. 2842–2846.
26. Li, B.; Lin, H.; Zheng, L.; Sun, S.; Yin, Z. An open CNC system based on EtherCAT network. In Proceedings of the 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Xi'an, China, 3–5 October 2016; pp. 795–801.
27. Chuang, W.L.; Yeh, M.H.; Yeh, Y.L. Develop Real-Time Robot Control Architecture Using Robot Operating System and EtherCAT. *Actuators* **2021**, *10*, 141. [CrossRef]
28. Delgado, R.; Kim, S.Y.; You, B.J.; Choi, B.W. An EtherCAT-based real-time motion control system in mobile robot application. In Proceedings of the 2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), Xi'an, China, 19–22 August 2016; pp. 710–715.
29. Yoon, D.K.; Kim, S.Y.; Cho, J.; Lee, K.K.; You, B.J. Development of a compact motor controller supporting EtherCAT for a dual-arm telepresence robot. In Proceedings of the 2014 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), Kuala Lumpur, Malaysia, 12–15 November 2014; pp. 253–256.
30. Liu, Z.; Liu, N.; Zhang, T.; Cui, L.; Li, H. EtherCAT based robot modular joint controller. In Proceedings of the 2015 IEEE International Conference on Information and Automation, Lijiang, China, 8–10 August 2015; pp. 1708–1713.
31. Zhang, G.; Ni, F.; Li, Z.; Liu, H. A Control System Design for 7-DoF Light-weight Robot based on EtherCAT Bus. In Proceedings of the 2018 IEEE International Conference on Mechatronics and Automation (ICMA), Changchun, China, 5–8 August 2018; pp. 2169–2174.
32. Delgado, R.; Park, J.; Lee, C.; Choi, B.W. Safe and Policy Oriented Secure Android-Based Industrial Embedded Control System. *Appl. Sci.* **2020**, *10*, 2796. [CrossRef]
33. Pan, C.T.; Sun, P.Y.; Li, H.J.; Hsieh, C.H.; Hoe, Z.Y.; Shiue, Y.L. Development of Multi-Axis Crank Linkage Motion System for Synchronized Flight Simulation with VR Immersion. *Appl. Sci.* **2021**, *11*, 3596. [CrossRef]
34. Cena, G.; Bertolotti, I.C.; Scanzio, S.; Valenzano, A.; Zunino, C. Evaluation of EtherCAT distributed clock performance. *IEEE Trans. Ind. Inf.* **2011**, *8*, 20–29. [CrossRef]
35. Park, S.M.; Kim, H.; Kim, H.W.; Cho, C.N.; Choi, J.Y. Synchronization improvement of distributed clocks in EtherCAT networks. *IEEE Commun. Lett.* **2017**, *21*, 1277–1280. [CrossRef]
36. Park, S.M.; Kwon, Y.; Choi, J.Y. Time Synchronization Between EtherCAT Network and External Processor. *IEEE Commun. Lett.* **2020**, *25*, 103–107. [CrossRef]
37. Park, S.M.; Kim, H.W.; Kim, H.J.; Choi, J.Y. Accuracy improvement of master–slave synchronization in EtherCAT networks. *IEEE Access* **2020**, *8*, 58620–58628. [CrossRef]
38. Kim, I.; Kim, T. Guaranteeing isochronous control of networked motion control systems using phase offset adjustment. *Sensors* **2015**, *15*, 13945–13965. [CrossRef]
39. Beckhoff Automation GmbH. *EtherCAT System Documentation*; Beckhoff Automation: Verl, Germany, 2020.
40. Seoane, L.; Diaz, C.; Zafra, J.; Ibarmia, S.; Quintana, C.; Pérez, C.; Moral, A.; Araujo, A. CAN implementation and performance for Raman Laser Spectrometer (RLS) Instrument on Exomars 2020 Mission. *IEEE Trans. Emerg. Top. Comput.* **2018**, *9*, 67–77. [CrossRef]
41. *Standard IEC 61800-7-1:2015*; Adjustable Speed Electrical Power Drive Systems-Part 7-1: Generic Interface and Use of Profiles for Power Drive Systems-Interface Definition. International Electrotechnical Commission: Geneva, Switzerland, 2015.
42. Felser, M. Real-time ethernet-industry prospective. *Proc. IEEE* **2005**, *93*, 1118–1129. [CrossRef]