Motor controller designed for robotics based on microcontroller with integrated EtherCAT

Conference Paper · May 2018

DOI: 10.1109/CarpathianCC.2018.8399643

2 authors, including:

Vlastimil Setka
University of West Bohemia
9 PUBLICATIONS **11** CITATIONS

Motor Controller Designed for Robotics Based on Microcontroller with Integrated EtherCAT

Vlastimil Šetka, David Tolar
NTIS – New Technologies for the Information Society Research Center,
Faculty of Applied Sciences, University of West Bohemia, Technicka 8, Pilsen, Czech Republic
E-mail: {setka,dtolar}@ntis.zcu.cz

***Abstract*—The paper focuses on our servo motor controller (SMC) design optimized for robotic applications. This controller is designed to be embedded inside robotic arms or joints to eliminate extensive cabling needed with external centralized controllers. It is based on a single Infineon XMC4800 microcontroller with integrated EtherCAT Slave Controller (ESC), powerful ARM Cortex-M4 CPU core and many peripherals suitable for motor control. This integrated solution provides precise control over distributed system timing.**

The hardware design contains two-axis three-phase inverter power stage with sigma-delta ADCs for high quality phase current measurement, interfaces for quadrature encoders or hall sensors and encoders with high-speed serial communication protocols and control part with a single microcontroller.

The paper provides description of hardware design, software architecture spanning from low level parts to velocity and position control loops with code generated from user reconfigurable function block diagrams and real-time performance evaluation.

***Index Terms*—Servo Drive, Motion Control, Servomotors, Industrial Robots, Real-time Systems, EtherCAT, REXYGEN**

I. Introduction

Servo motor control is an essential part of various robotic manipulators design. Typical examples of such manipulators are 6-DoF robotic arms, widely used in todays industrial manufacturing, and available as a standard commercial products from many vendors. Our research group[1] recently designed several types of non-standard manipulators like NDT (Non Destructive Testing) manipulator for inspecting the welds of the pipes of complex geometries in the field [1], or underwater vehicle for inspections [3]. Currently we are working on a new "ROBIN" (ROBotic INspection) manipulator for pipe welds inspections which combines tooth-belt driven motion around pipe [2] with 6-DoF conventional stage (see Fig. 1) and another stage consisting of three 2-DoF lightweight joints with a small tooth-belt driven actuators [4] to provide required level of flexibility.

All the application examples outlined above are driven by electric motors with a precise position feedback usually provided by encoders (angular position sensors). In the last mentioned case, there will be 14 motors on a single manipulator working in coordinated motion. Usualy these are BLDC (brushless direct current, with electronic commutation) motors or PMSMs (permanent magnet synchronous motors) or simple DC (direct current, with brushed commutator) motors rated at 12 to 60 volts and 10 to 400 watts. These parametes defines our area of interest.

Basic position feedback can be provided by hall-efect sensors embedded in the motor (usualy used for electronic commutation). Various types of encoders working on optical or magnetic principle, incremental or absolute, could be used for a precise position control. Interfaces could be a simple quadrature signal or a digital data communication protocols like BiSS [15]. Motors are typically combined with gearboxes which are not a perfect components – they suffer from backlash, friction and elasticity. That's why using two encodes could be required in the high-end applications – one on the motor side and another on the gearbox output – to compensate such imperfections by control algorithm.

For servo motor control, specialized electronic units called Servo Motor Controllers (SMC), or Servo Drives (SD), are used. Such SMC integrates power stage, encoder input and digital processing of torque, velocity, and position feedback control loops. Trajectory planning and kinematic model of manipulator are implemented in central control system, and position setpoints (sometimes with velocity and torque feedforwards) are cyclically sent to SMCs for each axis (motor) by filedbus systems like CAN or EtherCAT [13]. Typical communication period is between 8 ms and 0.25 ms.

This paper presents our design of two axis motor controller developed for applications described above. The design addresses our specific needs for high-quality control of multi-axis serial robotic manipulators and research on related algorithms, especially integration of multiple sensors, deployment of specialized control algorithms and cross-axis MIMO control.

The paper is organized as follows. Section II discusses why to develop custom solution instead of the commercially available ones. Section III deals with overall architecture, hardware design based on single microcontroller, phase currents measurement technique and system timing. Section IV briefly describes software implementaion and approach to user-defined control algorithms based on code generation from function block diagrams. Section V provides results of real-time performance tests. Finally, Section VI brings conclusions and directions for future work.

[1]Machine and processes control systems group, NTIS Research Center, University of West Bohemia in Pilsen, CZ, http://ntis.zcu.cz/en/

II. Motivation and Problem Statement

Although there are dozens of vendors providing COTS (commercial off the shelf) SMCs on the market, it's often problematic to use such products in a specialized manipulator like the "ROBIN" described in the introductory section. To name a few problems:

Dimensions and embedding – In case of large number of motors, especially on serial manipulator, it's impractical (if not impossible) to track all the cables from motors and sensors to some central control cabinet. Instead of, motor controllers are placed next to the motors – embedded inside the manipulator. Although some small embeddable SMC modules are available on the market, they more or less suffer from other limitations.

Control algorithms – SMC's control structure is usually fixed cascade of torque/velocity/position controller with some basic filters. Set of predefined parameters can be used to tune regulators and filters or switch some structure branches. But more complicated applications can profit from customized structure, utilizing multiple encoders, additional sensors like torque-cells or accelerometers for vibration damping, special filters and system models. Defining custom control algorithms at the velocity and position loop level is not a standard feature of commercial SMCs, especially those suitable for embeddeding. Even if standard structure fits to the application, user is often facing poorly documented behavior of many parameters.

Interfaces and performance – Interfaces for additional sensors mentioned in previous paragraph are often missing or do not provide sufficient performance. EtherCAT communication with central controller, which can provide really good performance, is getting to be de-facto standard, but there are various implementation-specific performance limits. Communication cycle about 1 ms, which is a common minimal standard accros vendors, is just enough for trajectory following. But complex machines are MIMO (multiple inputs, multiple outputs) systems. Advanced control algorithms for such systems lead to cross-axis control loops, and then we need communication of process data between motor controllers over EtherCAT with latency as low as possible – shorter than velocity controller cycle time in optimal case. This is not a common standard. Another limits apper if you want to map many objects into process data. For a fine tuning, continuous tracing of multiple internal signals with full sampling rate (e.g. oversampling) is needed, but this is also absolutely not a standard feature.

The issues described above led us to decide to develop a new custom motor controller.

III. Overall Architecture and Hardware

A. Requirements

We've already developed less sophisticated motor controller, and successfully use it in several applications, see our previous paper [5]. The experience with it and needs of several applications like the "ROBIN" lead to the following requirements.

Major improvements are preference of the EtherCAT communication [13] over CAN, which provides 100 times higher throughput (100 Mb/s vs. 1 Mb/s), two-axis in one controller, and significantly faster sampling rates. Requirements list:

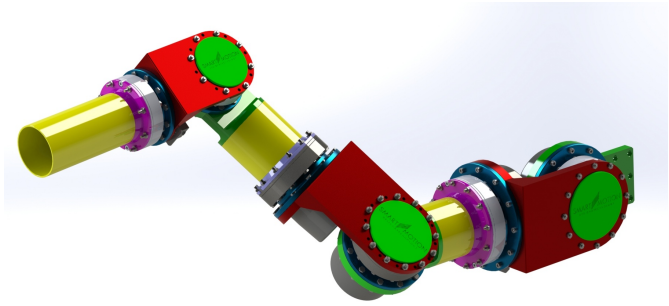

Fig. 1. An example of our motor controller use case – the "ROBIN" manipulator 6-DoF serial arm.

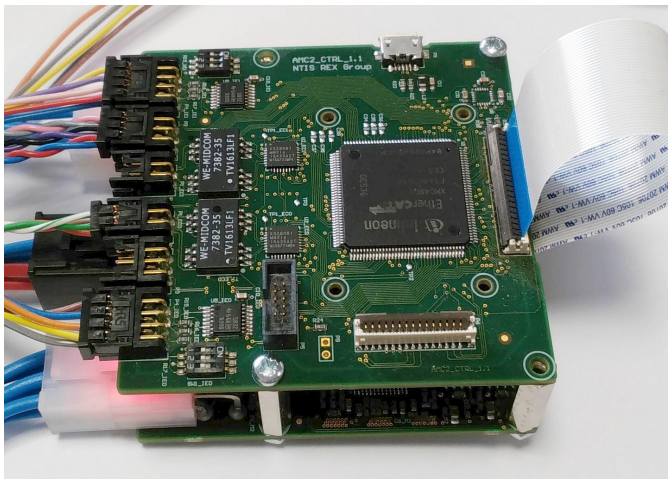

Fig. 2. Motor controller prototype, overall dimensions 65×65×25 mm.

- Compact design to fit inside the manipulator tube. Two motors per controller for easy cross-axis control.
- Daisy-chain connection for wiring controllers in series.
- Support for BLDC, PMSM motor with hall sensors, DC motors. Up to 48 V DC power, 3 A continuous, 6 A peak. Modular design for easy upgrade to a higer power.
- Support for BiSS serial protocol encoders (one per axis), optional quadrature encoders.
- EtherCAT communication with minimal overhead and processing times to allow low-latency data exchange between controllers.
- Low-noise phase current measurements based on $\Sigma\Delta$ (sigma-delta) ADCs (Analog to Digital Converters).
- Extension interface for optional I/O modules (analog inputs, additional encoders, CAN bus bridge, ...).
- PWM frequency and torque control loop at $\geq$ 20 kHz, velocity and position control loops at $\geq$ 8 kHz. CPU with enough computing power for algorithms more complex than standard cascade PID control at these sampling rates.

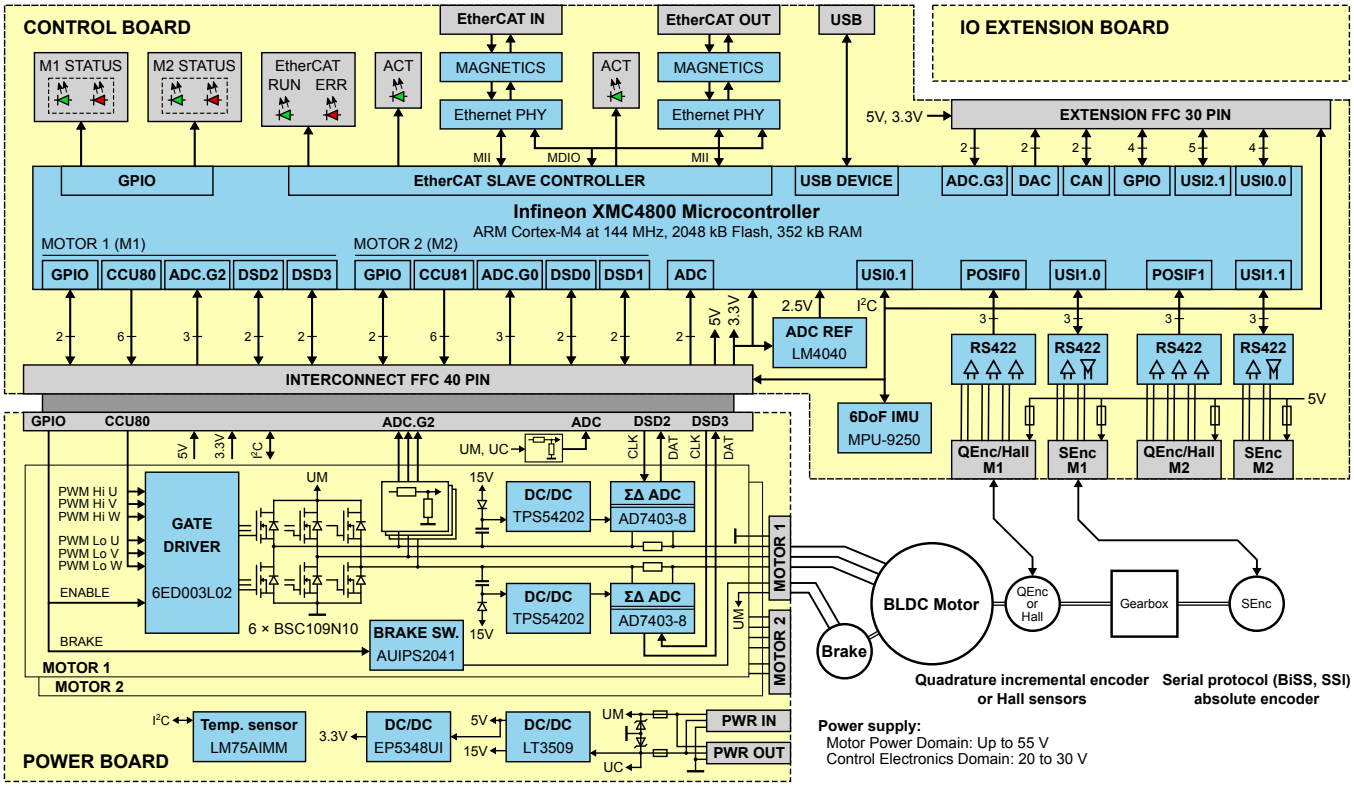

Fig. 3. Two axis motor controller hardware block diagram, consisting of the Control board, Power board, and optional IO extension board.

B. Hardware Design

The Infineon XMC4800 microcontroller (MCU) [11] was chosen for implementation, as this is one of the very few microcontrollers on the market with integrated EtherCAT Slave Controller (ESC). It also has all the peripherals needed for motor control, including ΣΔ demodulators, and relatively powerful ARM Cortex-M4 core running at 144 MHz. Embedded 2 MB of program FLASH memory and 352 kB of RAM is just enough for intended tasks. Direct integration of ESC in microcontroller should provide as low as possible latency of process data manipulation.

Overall detailed hardware block diagram is in the Fig. 3, prototype photo in the Fig. 2. Electronics is split into:

- *Control Board* – Contains MCU, EtherCAT input and output port, diagnostic LEDs, dual input for hall sensors or quadrature encoder (utilising POSIF units in the MCU), dual interface for serial communication encoder, USB device port for service communication and also Inertial Measurement Unit chip (IMU, 3 axis MEMS gyroscope + accelerometer + compass).
- *Power Board* – Connected to Control Board by a flat flexible cable. Contains DC/DC converters generating internal power rails (15 V, 5 V, 3.3 V) from input 24 V, dual power stage with gate driver and three half-bridges, current measurement on two phases using low resistance shunts and isolated ΣΔ modulators (ADCs, will be elaborated more in the next chapter) which each needs another DC/DC converter to generate 5 V power rail floating above its phase wire. Voltage on phase wires and input voltages are measured by the MCU integrated ADCs. There are two power connectors for daisy-chain wiring. Power supply is splitted between control domain (20 to 30 V) and motor domain (up to 55 V) which keeps control domain working when motor voltage is disconnected for safety reasons.
- *IO Extension Board* – Optional, can be connected to Control Board by a flat flexible cable. Several MCU peripherals are routed to extension connector – two ADC and two DAC channels, one CAN interface, two universal serial channels (can be configured as UART, SPI, I2C), some GPIOs and power rails.

C. Phase Current Measurement Using ΣΔ ADCs

One of the most important problems in motor control is phase current measurement. Due to PWM (Pulse Width Modulation) switching, phase currents contain significant noise. On the other hand, current control feedback loop should be calculated on the same (or even twice) frequency as the PWM, so usual low-pass filters can not be used. This is usually solved by synchronous sampling using a fast SAR (successive approximation) ADCs at a precise time instants in the middle between PWM switching times, when there is the least chance to catch transients. This technique is obviously not so much robust and limits current control performance and precision.

Another problem is current sensing circuicity, because phase wires floats between the ground and full voltage. This could be solved by isolated hall-efect transducers or analog isolators (which influence precision, especially at low current levels) or special non-isolated amplifiers (with limited voltage range and other problems).

These issues and possible solution using $\Sigma\Delta$ ADCs were elaborated in many papers, for example [10] or series [6] [7] [8] [9] which introduced several advanced techniques for increasing current control bandwidth.

Basic principle of $\Sigma\Delta$ analog to digital conversion is continuous sampling at very high frequency (MHz range) with a single bit output. This single bit digital signal, together with clock, can be easily isolated to solve circuicity problem outlined above. We used AD7403 [12] integrated $\Sigma\Delta$ modulator with integrated digital isolators, clocked at $f_S = 18$ MHz which is the highest integral fraction of 144 MHz MCU clock below the 20 MHz limit. Input range is ± 250 mV which allows us to use only 25 mΩ shunts for 10 A range.

The single bit output stream from $\Sigma\Delta$ modulator is then filtered and decimated. The most common for this task is a 3rd order sinc filter (sinc3) for its easy implementation and response characteristics (see [9]). XMC4800 MCU implements a quad channel CIC3 (cascaded integrator-comb) filter which is a different form of sinc3. The only parametr of CIC3 filter is Decimation Rate (DR). Filter produces sub-samples with rate $f_S \cdot DR^{-1}$, and its magnitude response has significant notches at frequencies $n \cdot f_S \cdot DR^{-1}$. Filter response is finite, symmetrical over 3 output samples ($3 \cdot DR - 2$ input clocks).

Based on a previous description, it is clear that DR has to be chosen that output sub-sampling rate is integer multiple of PWM frequency. Then the filter notches align with PWM noise spectrum. In our case: $f_{PWM} = 24$ kHz; $DR = 75$; $f_{ADC} = f_S \cdot DR^{-1} = 240$ kHz. So we get exactly 10 ADC sub-samples per PWM period. Following processing is defined by a complex timing which is elaborated in the next section.

Multiple filter chains can be used to get more precise (higher delay) and low delay (low precise) values at the same time. This can be used for overcurrent protection. XMC4800 MCU supports this as a 2nd auxiliary filter with configurable hardware trigger limits. We use it with $DR = 6$ which leads to only 0.5 us overcurrent detection delay.

D. System Timing Design

Many processes run both in parallel and in series inside a motor controller with complexity described above. So the system timing has to be planned carefully. Overall timing design of the most important processes is outlined in Fig. 4.

The main reference is PWM waveform, we choosed: $f_{PWM} = 24$ kHz, $T_{PWM} = 41.\overline{6}\ \mu s$. The point where PWM counter is at the maximal value is called *PWM Sync*, and a new PWM duty has to be calculated before it. PWM duty can be updated also at zero value of counter, e.g. twice per period, but we do not use this. All measurements in the system (phase currents, voltages, encoders positions) should be latched at the *PWM Sync* point or with a defined shift to it.

Since all control tasks should be synchronous with PWM to have fixed delay in control loops, and control loops in all motor controllers in the whole system should be synchronous to precisely follow coordinated motion commands generated from central controller, we have to slightly adjust the PWM period to keep everything in sync. The EtherCAT provides native mechanism called *Distributed Clock* (DC), which can be used to generate events synchronous on all slave devices in network with only tens of nanoseconds jitter. We use the *SYNC0* event and align it to the *PWM Sync* by a phase locked loop adjusting PWM period. The *SYNC0* is configured to trigger at the same period as EtherCAT communication, which should be multiple of T_{PWM}, typical value is 1 ms.

- $\Sigma\Delta$ *ADCs* – The PWM adjustments to the DC does not allow to run ADCs filters continuously aligned. Instead of, we start ADCs half cycle after center point between the two *PWM Sync* events, and stop it after 9 sub-samples which is a half cycle before the following center point. The first two samples has to be skiped for filter settling and next seven samples are accumulated, which is effectively a sinc1 filter. Magnitude response of such structure is center aligned around the *PWM Sync* point.
- *Serial encoder, BiSS protocol* – The first clock latches position, so trasfer is started at *PWM Sync*. Result should be ready before *Motor Loop* start if encoder is used for commutation or Field Oriented Control (FOC). With 4 MHz clock and 20 bits encoder data, this needs about 12 us, well before last ADC sub-sample.
- *Motor Loop* – The last ADC sub-sample triggers motor control (BLDC/FOC) and current controler task for both motors, it also re-initialize ADCs for next period. It has to provide new PWM duty values before next *PWM Sync*.
- *Application Loop* – Contains velocity and position controller and user-defined control code. It is triggered at the end of each 3rd *Motor Loop*, so it runs with 8 kHz rate.
- *EtherCAT RxPDO* – Copy setpoints from EtherCAT slave memory to *Application Loop* or even *Motor Loop*. Triggered by *SYNC0*, which is aligned to *PWM Sync*.
- *EtherCAT TxPDO* – Copy actual data to EtherCAT slave memory. Triggered by n-th (given by *Input Shift Time*, default 1st) *Motor Loop* completion after *SYNC0*.
- *EtherCAT Frame* – Should arrive not before *TxPDO* completion, and should end not after next *SYNC0*.

IV. Software Design

Low level software is implemented in a C language using DAVE development environment provided by Infineon. DAVE provides also libraries and configuration tools for MCU peripherals. The Slave Stack Code (SSC) provided by EtherCAT Technology Group [13] is used as a basis for EtherCAT device.

The low level part is coded by hand and includes peripheral initialization, motor control strategies and current control loop, protective functions, basic state management according the CiA402 standard [14] and EtherCAT integration. We follow and extend our approach described in a previous paper [5].

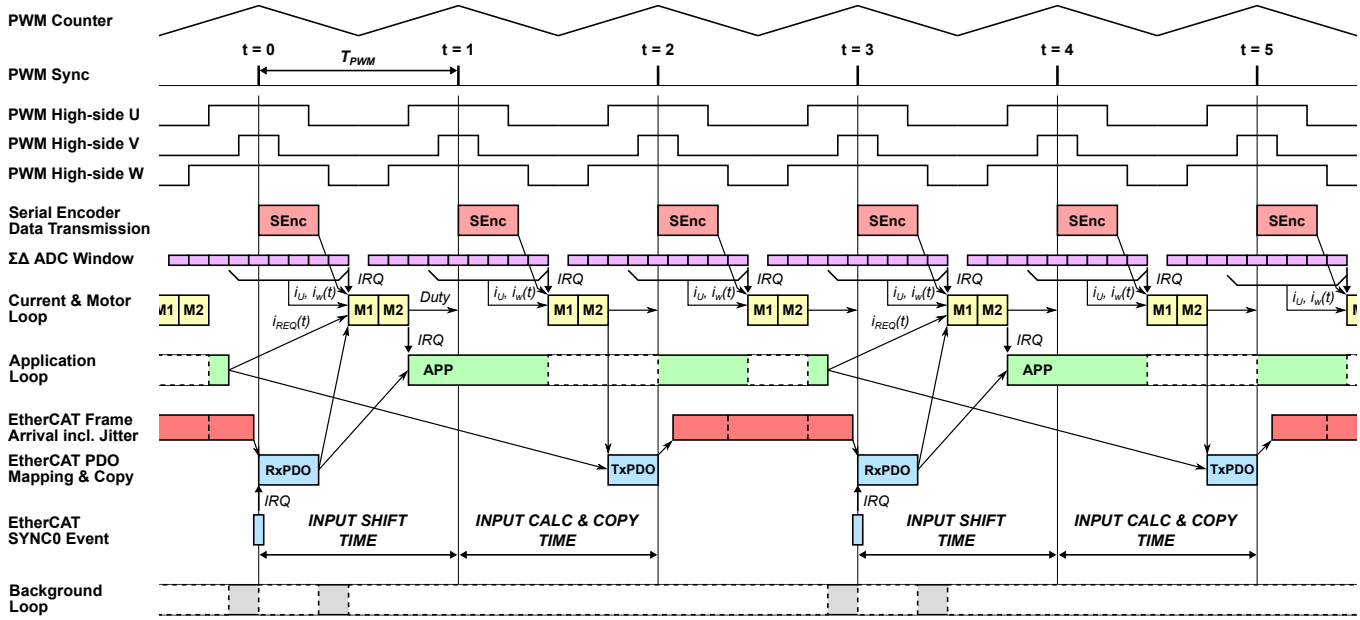

Fig. 4. Motor controller system timing diagram.

Custom code generation technology is used in two ways. Application algorithms (APP) including velocity and position control loops are designed as a fuction block diagrams using the RexLib block library [17] and REXYGEN [16] [18] design and simulation software tools. Thanks to this technology, control engineers can easily adjust and extend application algoriths with full understanding of function blocks behavior. Standard block diagram for one axis can be seen in Fig. 5. Common API is defined for APP with several categories:

- `act` – actual local measured values, read into APP,
- `req` – reqested values for local controllers (setpoint for current controller), write from APP,
- `in` – input data from EtherCAT, read into APP,
- `out` – output data to EtherCAT, write from APP.

Another important software subsystem, which can be hardly managed by hand coding, is parameters management. Typical SMC can specify hundreds of parameters which needs to be monitored and changed during machine commissioning and tuning. We define all parameters in Open Document Spreadsheet table which contains: ID, name, type (including enum list and bitfields structures), pointer to data, pointer to change flag, index and subindex for CANopen-over-EtherCAT, default value, persistent flag, read/write flag, access level, description. At compile time, C code with descriptor table and fragmens of EtherCAT Slave Information data are generated from this spreadsheet. Generic infrastructure is implemented for parameters storage in persistent memory and access over EtherCAT. Selected signals and block parameters from application block diagram are automatically included, so they can be read and written asynchronously, mapped to EtherCAT PDO and monitored by the data tracing subsystem.

The same spreadsheet is then loaded by our *Configurator* diagnostics application, which provides online access to all parameters with a convenient user interface and additional functions like backup/restore. *Configurator* exchanges data with EtherCAT Master, in our case the REXYGEN [16], over HTTP REST protocol.

V. Performance Evaluation

To evaluate system timing on current software implementation state and verify original requirements and expectations, basic real-time tests were performed. The most important information is how long CPU time is needed for individual tasks. The measurements were done using several GPIO pins which were set/reset on beginning/end of the task. Ouput waveforms were anlyzed by digital oscilloscope with logic analyzer. CPU time is significantly affected by selected level of compiler optimizations, so the critical tasks were measured on several levels. Results are summarized in Table I.

TABLE I
Real-time performance evaluation of motor controller

Task		run time [us] min	max	avg
Motor control - two axis	O0	21.22	21.96	21.56
(ADC processing, BLDC, current PID)	O2	6.62	7.27	6.67
Application task - two axis	O0	54.62	55.28	54.80
(velocity and position PID)	O2	21.26	23.02	21.82
	O3	19.78	21.44	20.28
Encoder communication (4 MHz clock)	-	11.44	11.44	11.44
Encoder data processing	O2	0.84	0.94	0.88
EtherCAT copy 128 bytes to ESC	O2	30.84	30.92	30.88
EtherCAT copy 128 bytes from ESC	O2	41.30	41.36	41.34

Generally, O0 (no optimizations) to O2 (optimize more) step provides speedup to about 40 to 30 % time. Step from O2 to O3 (optimize most) is to about 90 % time. Deviation between minimal and maximal times is under 10 % which is acceptable.

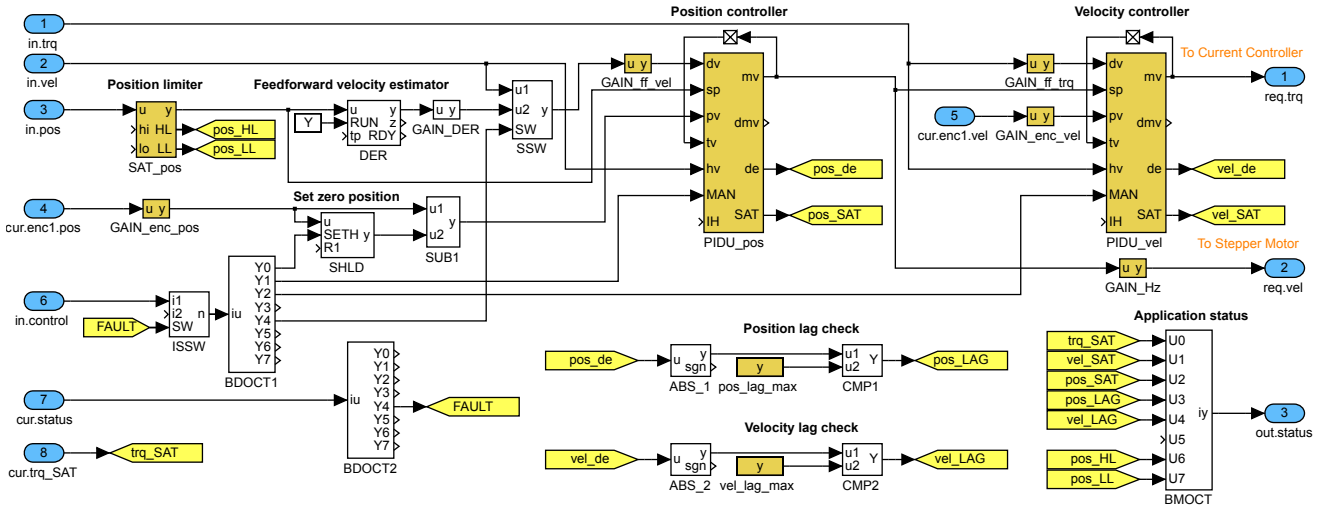

Fig. 5. Basic application algorithm implemented as a REXYGEN block diagram, contains velocity and position control loops based on the two PIDU blocks.

At O2 optimizations level, all control and service and standard application algorithms for two motors could run even at 24 kHz, which is over original expectations. But EtherCAT data copying at 128 bytes size takes about 31 us (TxPDO) and 41 us (RxPDO) which is much worse than expected and should be analyzed more in-depth. A quick research using debugger shows that memcpy() copying is done in byte-by-byte loop for some reason, but this explains the times and the difference between TxPDO and RxPDO only partially.

VI. Conclusion and Future Work

This paper presented hardware and software design of a two-axis servo motor controller for application in robotic manipulators. Thanks to a modern microcontroller with integrated EtherCAT Slave Controller, final control board solution is almost single-chip and provides enough computational performance to run standard control structure – cascade of torque/velocity/position controller – for both axes at 24 kHz sampling rate. The EtherCAT data manipulation timing is slower than required for a complex scenarios which will need more research, tuning and optimizations.

The main goal – to overcome limitations of comercially available motor controllers – was archieved. Mainly thanks to application and further improvement of technology for control algorithm code generation from function block diagrams proven in our previous work. Further effort should provide convenient way to downloading a new control algorithms over EtherCAT during runtime and support for more motor types.

Acknowledgment

This work was supported by the project LO1506 of the Czech Ministry of Education, Youth and Sports, and the Technology Agency of the Czech Republic under Grant TF02000041.

References

[1] M. Švejda, T. Čechura. “Interpolation method for robot trajectory planning”. *2015 20th International Conference on Process Control (PC)*, Strbske Pleso, 2015, pp. 406-411. doi: 10.1109/PC.2015.7169997

[2] M. Švejda. “New Robotic Architecture for NDT Applications”. *World Congress IFAC*, vol. 19, 2014, pp. 11 761–11 766.

[3] L. Bláha, M. Schlegel, J. Königsmarková. “Design and Control of Underwater Vehicle for NDT Inspections”. *2014 IEEE/OES Autonomous Underwater Vehicles (AUV)*, Oxford, MS, 2014, pp. 1-6. doi: 10.1109/AUV.2014.7054427

[4] L. Bláha. “Design of a compact 2-DOF joint with belt driven actuators”. *2017 IEEE 4th International Conference on Soft Computing & Machine Intelligence (ISCMI)*, Mauritius, 2017, pp. 198-202. doi: 10.1109/ISCMI.2017.8279626

[5] V. Šetka, M. Štětina. “Software Tools for Embedded Reconfigurable Control Algorithm Code Generation”. *2016 17th International Carpathian Control Conference (ICCC)*, Tatranska Lomnica, 2016, pp. 711-716. doi: 10.1109/CarpathianCC.2016.7501188

[6] W. Peters et al. “Regular-Sampled Current Measurement in AC Drives Using Delta-Sigma Modulators”. *2009 13th European Conference on Power Electronics and Applications*, Barcelona, 2009, pp. 1-9.

[7] C. Klarenbach, J. O. Krah. “Fast and High Precision Motor Control for High Performance Servo Drives”. *PCIM Power Conversion Intelligent Motion*, Nürnberg, 2010, pp. 326-333.

[8] C. Klarenbach, H. Schmirgel, J. O. Krah. “Design of Fast and Robust Current Controllers for Servo Drives based on Space Vector Modulation”. *PCIM Power Conversion Intelligent Motion*, Nürnberg, 2011, pp. 182-188.

[9] A. Rath, C. Klarenbach, O. D. Djouosseu, J. O. Krah. “Fast Current Measurement based on Enhanced ΣΔ Technology”. *PCIM Power Conversion Intelligent Motion*, Nürnberg, 2012, pp. 258-266.

[10] J. Sorensen. “Sigma-Delta Conversion Used For Motor Control”. *Analog Devices Technical Article*, 2012. http://www.analog.com/en/technical-articles/sigma-delta-conversion-used-for-motor-control.html.

[11] Infineon Technologies AG. “XMC4700 / XMC4800 Datasheet – Microcontroller Series for Industrial Applications”, V1.0, Edition 2016-01.

[12] Analog Devices. “AD7403 Datasheet – 16-Bit, Isolated Sigma-Delta Modulator”, Rev. B, 02/23/2017.

[13] EtherCAT Technology Group. “EtherCAT – the Ethernet Fieldbus”. [Online]. https://www.ethercat.org/en/technology.html

[14] CAN in Automation, CiA 402 series: CANopen device profile for drives and motion control. [Online]. http://www.can-cia.org/can-knowledge/canopen/cia402/

[15] BiSS Association e.V. i.G. “About BiSS” [Online]. http://biss-interface.com/about-biss/

[16] REXYGEN Software Tools. [Online]. https://www.rexygen.com

[17] REX Controls s.r.o. “Function Blocks of the REX Control System”. Version 2.50.5, 2017-09-06. [Online]. Available: https://www.rexcontrols.com/media/DOC/ENGLISH/BRef_ENG.pdf.

[18] P. Balda and M. Schlegel. “Advanced PID Control Algorithms Built into the REX Control System”. *IFAC Conference on Advances in PID Control*, Brescia, 2012.