Application of Ethernet Powerlink for communication in a Linux RTAI open CNC control system

Krystian Erwiński, Marcin Paprocki, Lech M. Grzesiak, *Senior Member, IEEE*, Kazimierz Karwowski, and Andrzej Wawrzak

***Abstract*—In Computerized Numerical Control (CNC) systems the communication bus between the controller and axis servo drives must offer high bandwidth, noise immunity and time determinism. More and more CNC systems use real-time Ethernet protocols such as Ethernet Powerlink (EPL). Many modern controllers are closed costly hardware-based solutions. In this article the implementation of EPL communication bus in a PC-based CNC system is presented. The CNC system includes a PC computer, software CNC controller running under Linux Real Time Application Interface (RTAI) Real-Time Operating System (RTOS) and servo-drives communicating via EPL. The EPL stack was implemented as a real-time kernel module. Due to software-only implementation this system is a cost-effective solution for a broad range of applications in machine control. All software is based on GNU General Public License (GPL) or Berkeley Software Distribution (BSD) licenses. Necessary modifications to the EPL stack, Linux configuration, computer BIOS and motherboard configuration were presented. Experimental results of EPL communication cycle jitter on 3 different PC's were presented. The results confirm good performance of the presented system.**

***Index Terms*—Computerized Numerical Control, Enhanced Machine Controller, Ethernet Powerlink, RTAI Linux.**

I. Introduction

Most CNC control systems can be subdivided into two main groups. One group contains standalone controllers, others utilize personal computers (PC's). Standalone controllers use dedicated embedded solutions such as microcontrollers [1], Field Programmable Gate Arrays (FPGA's) [2],[3],[4] and Digital Signal Processors (DSP's) [5], and others [6].

Manuscript received May 29, 2012. Accepted for publication , June 14, 2012.

This research was supported by the Nicolaus Copernicus University Faculty of Physics with grants for young researchers 400-F and 407-F.

K. E., M.P., K. K. and A.W. are with the Faculty of Physics, Astronomy and Informatics; Nicolaus Copernicus University, Torun, Poland (erwin@fizyka.umk.pl, zwirek@fizyka.umk.pl, kkarwowski@fizyka.umk.pl, awawrzak@fizyka.umk.pl).

L. M. G. is with the Institute of Control and Industrial Electronics, Warsaw University of Technology, Warsaw, Poland (l.grzesiak@isep.pw.edu.pl).

The CNC controllers implemented on the general-purpose PC usually utilize some form of dedicated hardware to achieve time determinism required for real-time control of the machine's drives. Such hardware includes peripherals such as ISA, PCI and PCI-E expansion cards, external FPGA [7] and DSP [5],[8] modules, timer cards [9], digital communication controllers [10]. In the aforementioned solutions the PC only handles the Human Machine Interface (HMI) part of the CNC control system and the dedicated hardware implements time critical motion control tasks.

Although CNC controllers implemented partially or completely as embedded systems can achieve high levels of determinism they are expensive, inflexible, closed solutions. Another way to implement a CNC control system is to use a PC with a RTOS connected to the machine's servo drives via fieldbus. The RTOS adds time determinism required for coordinated real-time control of the machine's servo drives without the need for dedicated hardware. RTOS'es often utilized for implementing a PC-based CNC control system include: µC/OS-II [11], Windows CE.NET [9], RTLinux [10], RTAI-Linux [12], and others [13],[5],[7].

There are many communication bus architectures used in Open CNC systems. If the system does not need a high bandwidth buses such as: CAN[8],[14] and others [3] are used. When high bandwidth is required, an Ethernet fieldbus should be used. Standard Ethernet does not offer determinism required for real-time motion control but there are many industrial Ethernet modifications available. Some are custom non-standardized modifications developed by various research groups such as: RTnet[12], and others [4],[15],[16],[17].

Major manufacturers of industrial control equipment have also released their own Ethernet-based fieldbus standards. Real-time Ethernet standards such as: ProfiNET [18],[19], Sercos III [20], EtherCAT [5],[21] or Ethernet Powerlink [21] are becoming more and more popular due to availability of supported products. Many servo drive manufacturers offer complete CNC control systems that utilize industrial Ethernet fieldbuses. These solutions are usually external embedded devices which utilize the PC as a HMI and implement a complete real-time numerical control kernel. Such a proprietary closed control system is costly and greatly limits adaptability to a particular application. It also makes it

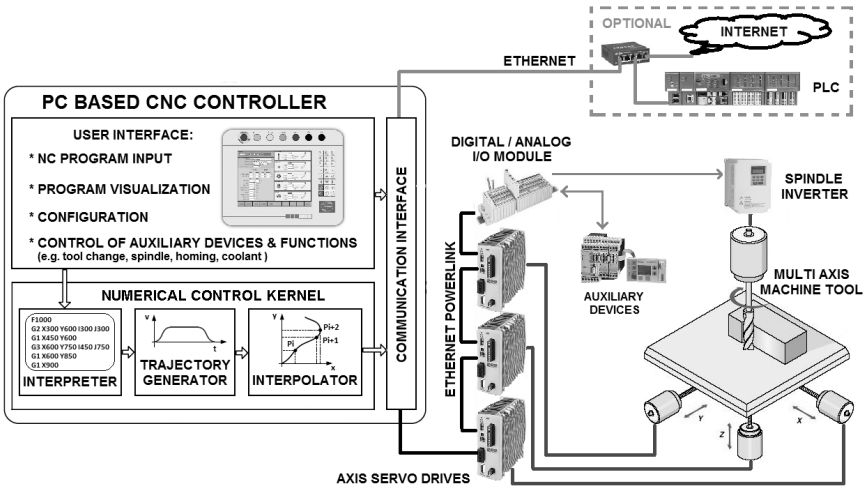

Fig. 1. Schematic of Ethernet Powerlink based CNC control system.

impossible to expand the numerical control kernel with new features required for certain applications.

In this article the architecture of a PC-based open CNC multi-axis machine control system with Ethernet Powerlink (EPL) is presented. The authors decided to adopt the PC-RTOS CNC controller structure due to its low cost (purely software solution without any expansion cards or external controllers), flexibility (ease of adding new functionality by expanding the PC-implemented numerical control kernel and HMI) and ease of implementation. The main goal behind developing this CNC control system was to achieve the best possible time determinism available to a purely software solution while at the same time maintain low cost and flexibility. Keeping these goals in mind the authors decided to adopt Linux RTAI [22] as the CNC controller RTOS. This RTOS is free, open-source and has a large software and user base while at the same time offering real-time performance comparable to commercial RTOS'es. Furthermore a complete feature-rich free open-source software CNC controller EMC2 [23] was already available for RTAI.

One serious problem with this solution was lack of a widely supported standardized industrial Ethernet stack which could cooperate with commercially available servo drives. This is important from a practical perspective because using an industrial standard Ethernet fieldbus increases flexibility and reduces costs. Otherwise expensive custom-built servo drives would have to be used.

The authors decided to extend the functionality of the RTAI/EMC2 real-time CNC control system with an Ethernet Powerlink servo drive communication module. Ethernet Powerlink was chosen as the fieldbus because it is a completely open widely adopted standard that offers excellent time determinism. Many of the biggest servo drive manufacturers offer products with an Ethernet Powerlink interface (e.g. B&R ACOPOS, Baldor MicroFlex e100, Parker Aries EPL).

The main contribution of this article is the implementation the Ethernet Powerlink stack in Linux RTAI real-time operating system and extending it with the Can in Automation (CiA) 402 device profile for servo drives [24] . The developed Powerlink based communications module was adapted to cooperate with the EMC2 CNC control software thereby forming a complete open low-cost PC-based CNC control system that is able to cooperate with a variety of commercially available servo drives.

In Chapter II the proposed Multi-Axis CNC Control System is described in detail. The cost-effective PC-based CNC Controller overview is presented. The Ethernet Powerlink communication standard applied to multi – axis machine control is described. In Chapter III the implementation details of EPL stack in the proposed CNC system are presented. In Chapter IV the performance of the presented CNC system is analyzed. The authors measured EPL cycle jitter for 3 different computer platforms. Similar tests were described in [25],[26]. In Chapter V conclusions are given.

II. Structure of the Multi-Axis CNC Control System

The proposed system is presented in Fig.1. It consists of a PC based CNC controller communicating over Ethernet Powerlink fieldbus, Ethernet Powerlink compatible commercial servo drives, Ethernet Powerlink Input/Output (I/O) module and auxiliary machine equipment. The system can be extended to communicate over a standard Ethernet TCP/IP protocol.

The authors have used Ethernet Powerlink as the communication protocol between the PC based controller and the servo drives. This protocol uses standard Ethernet PCI network interface card. A line network topology was used because of small number of nodes typically used in a CNC control system. The PC is the first node on the EPL network. Drives are daisy chained to one another by a two port Ethernet hub embedded in each drive. The system can communicate also by a standard Ethernet TCP/IP protocol via a separate network interface. This communication can be used with a wide range of devices which do not need real-time performance like Programmable Logic Controllers (PLC). Furthermore, this can be used for internet access. Necessary software modifications of the EPL stack in the PC based CNC controller are performed to provide a real-time performance.

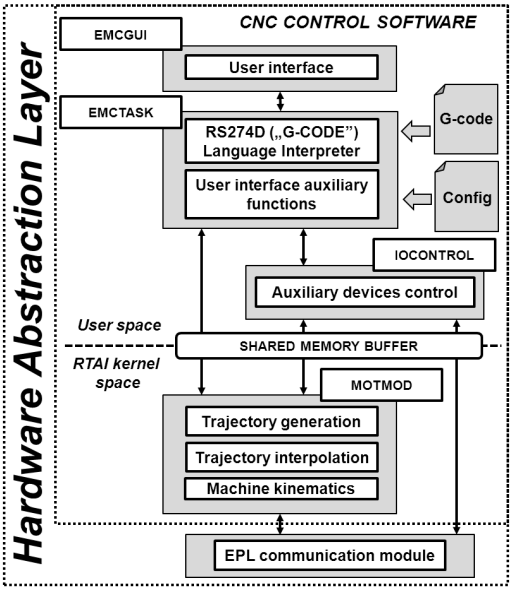

Fig. 2. Block schematic of the EMC2 CNC control application.

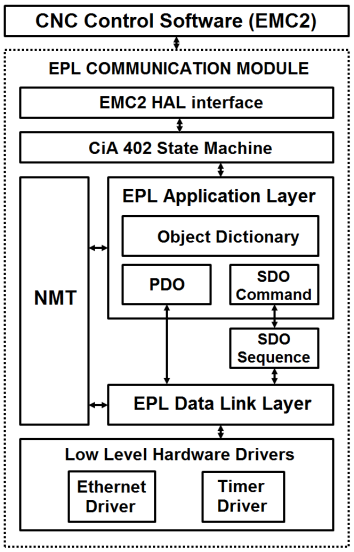

Fig. 3. EPL communication module.

Additionally, the authors have developed an Ethernet Powerlink compatible, I/O module. This device contains digital and analog inputs and outputs for controlling auxiliary machine tool devices like a spindle inverter, tool changer, limit and homing switches and so on.

A. PC based CNC controller

One of the problems in CNC machine tool control is sending position commands synchronously to servo drives. Ideally, all servo drives should receive and execute their respective position commands at exactly the same time. Also the periods between consecutive position commands should always be constant. In the presented system time interval length between new position commands sent to servo drives is assumed to be 1 millisecond. Furthermore execution of lower priority tasks (e.g. HMI task) must not affect the performance of high priority tasks. A real-time operating system is necessary to meet these constraints. Linux RTAI – a free open-source RTOS was used because it offers jitter and latency in the order of microseconds. This is comparable to commercial RTOS'es such as VxWorks or QNX. The block schematic of the EMC2 application is presented in Fig. 2. It consists of two kinds of modules – user space and RTAI kernel space [22]. User space modules are normal Linux programs that do not have a real-time performance.

EMCGUI is the top-level module providing the HMI for the application. This includes a full 3d viewer of the machined trajectory, software oscilloscope and special function controls (feed rate override, spindle control etc.). The user can also extend the Graphical User Interface (GUI) with his own controls if necessary. EMCTASK contains the "G-CODE" language interpreter [27]. It also controls additional functions of the CNC controller, loads configuration files. The EMCTASK module communicates with the user space I/O module IOCONTROL that controls the devices for instance limit and homing switches. The user space modules communicate with RTAI kernel space modules through a shared memory buffer.

RTAI kernel space modules operate in real-time because the RTAI nanokernel architecture [22] does not allow them to be preempted by Linux tasks and other real-time processes with a lower priority. It also intercepts and schedules all interrupts according to task priority.

MOTMOD is the main motion control real-time module. It contains trajectory generation and interpolation algorithms that compute commanded position values. Those values are determined from the trajectory contained in the shared memory buffer. MOTMOD also includes simple look-ahead functionality that allows the blending of sharp corners on the tool path. This is important for smooth machine operation, especially in high speed machining.

The EPL protocol stack was implemented by the authors in the form of a RTAI real-time module.

EMC2 uses its own API called the Hardware Abstraction Layer (HAL) for interfacing RTAI functionality. The HAL facilitates thread initialization and management, Inter-Process Communication (IPC), integration and management of different real-time modules into a single control system.

Inter-process communication between HAL modules is achieved via HAL pins. Pins are module variables exported to a common shared memory area. These variables are visible to all modules loaded into the HAL and can be used to easily connect different modules. Loaded modules and pin connections between them are defined in a config file.

B. EPL communication module

The EPL communication module is presented in Fig. 3. The module consists of low level hardware drivers, actual EPL protocol stack (including the Object Dictionary, EPL data link layer and network management state machine), supervisory state machine conformant with the CiA 402 device profile and an interface to the EMC2 HAL.

Standard Ethernet uses the CSMA/CD [28] multi access protocol for managing the network. Packet collisions can

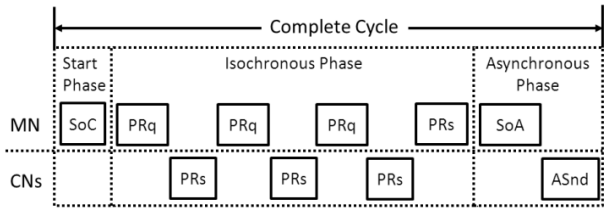

Fig. 4. Ethernet Powerlink communication cycle.

	Bit Offset								entry defined by
Octet Offset	7	6	5	4	3	2	1	0	
0 .. 5	Destination MAC Address								Ethernet II
6 .. 11	Source MAC Address								
12 .. 13	EtherType								
14	res	MessageType							Ethernet POWERLINK
15	Destination								
16	Source								
17 .. n	Data								
n+1 .. n+4	CRC32								Ethernet II

Fig. 5. Ethernet Powerlink basic frame format.[30].

occur in the case of multiple nodes transmitting at the same time. This protocol is inherently non-deterministic and cannot be used to directly control servo drives in CNC applications.

Ethernet Powerlink is an isochronous real-time communication protocol that meets high performance requirements in automation and motion control applications. It does not change the basic principles of the Fast Ethernet standard [28] but extends it with real-time capabilities.

Compared to the OSI model [29] of a standard Ethernet stack, only the physical layer (OSI layer 1) was left unchanged. The data link layer (OSI layer 2) was extended to include the real-time mechanism. A network layer (OSI layer 3) is not used. The rest of the layers have been replaced by Ethernet Powerlink ones. The standard physical and data link layer enable the use of standard Ethernet PC interfaces. The management of the EPL stack is performed by the Network Management (NMT) state machine.

The EPL network consists of one master Managing Node (MN) and up to 240 slave Controlled Nodes (CN) [21]. The EPL network may be of the line, star or tree topology type. The MN manages communication and configuration of the Ethernet Powerlink network.

Communication over an EPL network is performed in cycles. A single cycle consists of an isochronous phase and an asynchronous phase. The isochronous phase is divided into time slots – one per each CN. Only one node can transmit during each time slot. This means that collisions are avoided entirely and a high level of determinism is achieved. This bus scheduling mechanism is called Time Division Multiple Access (TDMA). During the asynchronous phase, the MN grants a time slot to a single CN or to itself.

The schematic of an Ethernet Powerlink communication cycle is presented in Fig.4[21]. The cycle starts with a "Start of Cyclic" (SoC) frame broadcast by the MN. This frame synchronizes all CNs with the MN. MN, then sends a "Poll Request" (PRq) frame to a single node. CN responds with a "Poll response" (PRs) frame containing data. This is repeated for every node on the EPL network. After all CN's have been serviced, the MN issues a broadcast PRs frame signalizing the end of the isochronous phase. In the asynchronous phase, MN sends a "Start of Asynchronous" (SoA) frame to all nodes. Data contained within this frame inform which node should respond to the MN request. Next, the chosen node sends an "Asynchronous Send" frame (ASnd) with proper data. The managing node waits in an idle phase until the cycle time has elapsed before sending another SoC to start a new cycle.

Ethernet Powerlink messages are encapsulated standard Ethernet frames. These messages contain 4 fields: message type (SoC, PRq, PRs, SoA, ASnd), destination node address, source node address, payload. The Powerlink basic frame format is presented in figure 5 [30].

The Ethernet Powerlink Application layer is based upon the CANOpen standard [30]. Its main element is the Object Dictionary (OBD). The OBD serves as an intermediary between the EPL stack and the CNC application (MN) or device firmware (CN). The object dictionary is a data structure containing all parameters relevant to the communication stack (e.g. cycle time, jitter tolerances, number of nodes, node configuration) as well as process variables sent between the MN and CN's. Every variable called an object has its own hexadecimal numerical identifier (index). The structure of the EPL OBD is presented in table 1.

TABLE I
THE STRUCTURE OF THE ETHERNET POWERLINK OBJECT DICTIONARY

Index	Object
0000h	Not used
0001h…001Fh	Static Data Types
0020h…003Fh	Complex Data Types
0040h…005Fh	Manufacturer Specific Complex Data Types
0060h…007Fh	Device Profile Specific Static Data Types
0080h…009Fh	Device Profile Specific Complex Data Types
00A0h…03FFh	Reserved for future use
0400h-041Fh	POWERLINK Specific Static Data Types
0420h-04FFh	POWERLINK Specific Complex Data Types
0500h…0FFFh	Reserved for future use
1000h…1FFFh	Communication Profile Area
2000h…5FFFh	Manufacturer Specific Profile Area
6000h…9FFFh	Standardized Device Profile Area
A000h…BFFFh	Standardized Interface Profile Area
C000h…FFFFh	Reserved for future use

OBD objects can be mapped to the Process Data Object (PDO) and Service Data Object (SDO) structures. Objects sent and received during every isochronous phase are mapped to the PDO. Those sent and received during the asynchronous phase are mapped to the SDO. Additionally, a SDO Sequence Layer is used to schedule sending SDO mapped objects when an asynchronous slot becomes available. Alternatively, SDO communication can be performed using a standard TCP/IP protocol instead of the Ethernet Powerlink protocol. Every node has its own OBD and writing values to the PDO/SDO mapped objects will update them in the corresponding objects on other devices on the network after transmitting them in the payload section of PRq or PRs frames.

Besides objects defined in the basic EPL standard the user can define their own objects in the appropriate OBD section. There are also sections of the OBD which are defined by separate standards for each type of controlled node. These standards known as device profiles define a set of objects

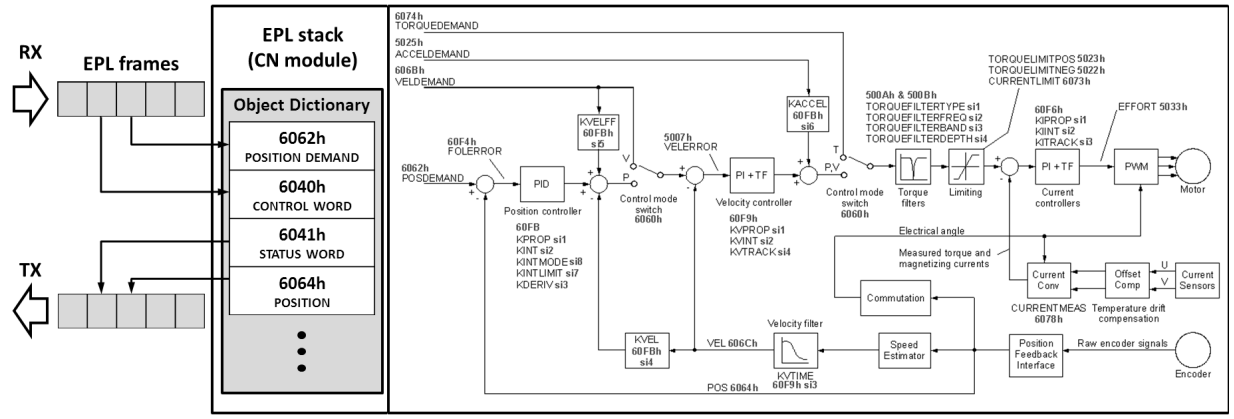

Fig. 6. Schematic of Baldor MicroFlex e100 drive control loop with EPL CIA402 object dictionary. Hexadecimal numbers above 6000 in control loop elements correspond to CiA402 objects. Numbers below 6000h are manufacturer specific objects.

containing process and configuration variables typical to each type of CN. The device profile for electrical drives is defined in the CiA402 standard. Every EPL compatible servo drive is required to support the same set of OBD objects with pre-defined addresses. These objects correspond to drive parameters such as position demand, actual position, speed or current measurement, digital I/O's, PID controller gains, drive operating mode (current, velocity or position) and so on. In the presented EPL communication module the OBD structure defined in CiA402 is fully supported. An example implementation of CiA402 OBD in the servo drive CN is shown in figure 6.

In addition to pre-defined objects the CiA402 standard also defines a state machine that controls operation of each servo drive. Current state of this state machine is indicated by the "Status Word" object and state change is triggered by the MN by modifying the "Control Word" object. Exchange of these two objects is required when controlling EPL servo drives. Appropriate state transitions triggered by the MN are required for proper drive initialization, operation and fault handling. This state machine is also implemented in the presented communication module.

Data exchange between the EPL module and the rest of EMC2 modules is performed via HAL pins. Each CiA402 object is mapped to a HAL pin. These pins are visible to the EMC2 MOTMOD motion control module and are written or read by it. A schematic presentation of this data exchange method is shown in figure 7.

The details of implementing EPL stack in the EMC2/RTAI real-time environment are presented in Chapter III.

III. Proposed solution of Ethernet Powerlink Implementation In EMC2 Environment

The Ethernet Powerlink stack is implemented as a Linux RTAI real-time module based upon the OpenPOWERLINK v1.6 source code [31]. Modification of the stack was necessary to incorporate real-time capabilities offered by Linux RTAI and support the CANOpen CiA402 standard.

The stack application layer was extended to include a CIA402 device profile for communication with servo drives. Object

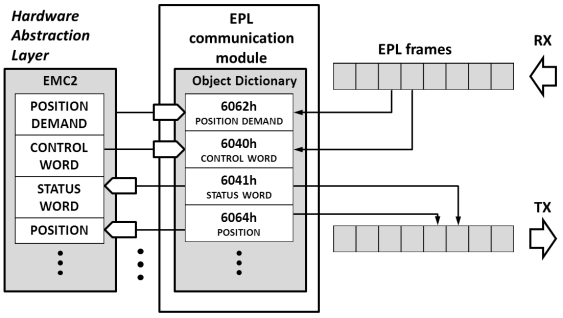

Fig. 7. Data exchange between EMC2 and EPL module via HAL pins

Dictionary entries containing process variables are sent in PRq and PRs EPL frames. OBD entries transmitted between the CNC controller and servo drives in the isochronous phase are presented in table 2.

TABLE II
PROCESS DATA TRANSMITTED BETWEEN THE CONTROLLER MN AND CN SERVO DRIVES IN THE ISOCHRONOUS PHASE OF THE POWERLINK CYCLE

PRq EPL frame payload	PRs EPL frame payload
Position Demand Value (6062h)	Actual Position Value (6064h)
Digital Outputs (60FEh)	Following Error Value (60F4h)
Modes of Operation (6060h)	Digital Inputs (60FDh)
Control Word (6040h)	Status Word (6041h)

In the asynchronous phase, configuration frames performing functions such as setting PID controller gains or identification of new drives is performed via ASnd frames sent by the MN. This configuration usually takes place at drive initialization time but configuration data can be updated later if necessary.

In order to adapt the OpenPOWERLINK v1.6 stack to the Linux RTAI real-time environment and ensure stable communication the authors had to introduce modifications to the EPL stack code as well as to Linux and PC BIOS configuration.

Following modifications were made to the stack:

- Shared memory buffers were replaced by direct function calls to improve performance.
- Linux kernel functions were replaced by their RTAI counterparts (memory allocation, spinlocks, atomic operations).

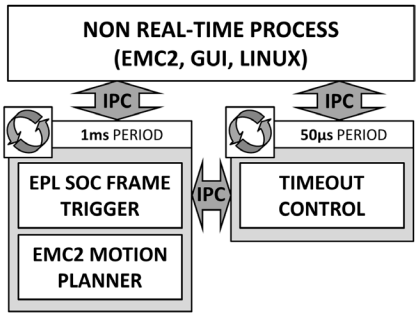

Fig. 8. Real-time threads used in the proposed control system.

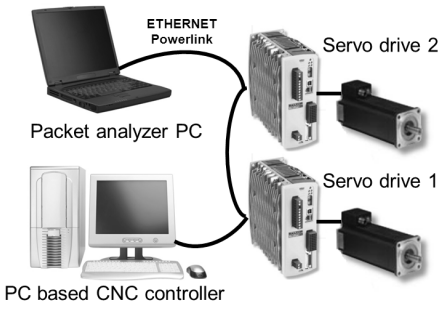

Fig. 9. The experimental station schematic.

- The network interface driver was modified to use RTAI interrupts. RTAI interrupt registration function is used instead of the Linux one so that the Ethernet driver interrupt is handled immediately.
- RTAI timers were utilized instead of Linux ones. Two HAL threads are utilized. One thread with a 1ms period triggers the start of an EPL cycle. This thread is shared with the trajectory planning and path interpolation module. Another thread (50µs period) is used for monitoring node response times in order to detect timeouts. Both threads utilize the Advanced Programmable Interrupt Controller (APIC) timer operating in periodic mode. The schematic representation of real-time threads used in the CNC control system is presented in figure 8.

Following changes were made in Linux configuration:

- Unnecessary devices and device drivers (e.g. sound card) were disabled.
- To ensure high performance, the network interface card was assigned an interrupt with as high priority as possible. This interrupt is not shared with other interrupts. If this is not possible for some computer configurations hardware associated with these shared interrupts should not be used.
- The real-time code (trajectory planner and EPL stack) was configured to run on a separate core isolated from the Linux scheduler via the ISOLCPUS boot parameter. Furthermore, the network card interrupt was assigned only to the real-time core via the IRQ affinity kernel system call. All other interrupts were assigned to the other cores to avoid interference with real-time operation. In order for the changes to be permanent interrupt balancing, which switches interrupts to different cores depending on each core's load, was disabled

Several BIOS settings were modified such as: disabling integrated sound card, power saving features, processor frequency scaling, thermal monitoring, dynamic fan speed regulation, legacy USB support and S.M.A.R.T. for hard drives. These functions use dedicated non-maskable interrupts that cannot be disabled by the operating system. When such interrupts are active during real-time operation they can cause unacceptable jitter and latency. Direct Memory Access (DMA) for hard drives was also disabled. This greatly decreases load placed on the system bus and therefore decreases jitter. Finally System Management Interrupts were disabled. These interrupts are used by modern mainboards for performing various tasks such as thermal throttling, system health checks, reporting hardware errors, power management etc. Disabling them required setting appropriate bits in chipset configuration registers.

IV. Experimental Setup and Test Results for Different PC Platforms

The EPL based CNC controller was implemented and tested on three different PCs. The experimental station is presented in figure 9. The hardware specifications of computers used in the experiment are presented in table 3. A separate computer (configuration identical to test computer 2) was connected to the network to capture and analyze frames with an RTAI based packet analyzer developed by the authors.

TABLE III
Hardware Specifications of The Test Computers

Components	1st computer	2nd computer	3rd computer
CPU	Core 2 Duo E7500 2.93GHz	Core 2 Quad Q8200 2.33GHz	Core i5 760 2.80GHz
Main Board (RAM)	IEI IMBA-9454ISA (2GB)	ASUS P5E Deluxe (4GB)	ASUS P7H55 (4GB)
GPU	Matrox G550 PCIe	ATI Radeon HD 4600	ATI Radeon HD 5670
Linux kernel	2.6.32-122-rtai	2.6.32-122-rtai	2.6.32-122-rtai
RTAI	3.8.1	3.8.1	3.8.1
PC APIC Timer	APIC Timer 16.667138MHz	APIC Timer 20.871775MHz	APIC Timer 8.362397MHz

The packet analyzer is a PC with RTAI running the EPL real-time low level driver. The driver was configured only to receive frames and measure time between receiving them. Measurement is done within the Ethernet card's interrupt handler. Time measurement was performed using the processor's Time Stamp Counter (TSC). The TSC is a 64bit counter running at the processor frequency (2.33GHz) therefore it is very accurate. There is one TSC per core. The reading of the TSC is performed by RDTSC assembly instruction which is very fast and introduces negligible overhead. The real-time code of the packet analyzer was locked to one processor core with a corresponding TSC. The packet analyzer also computes on-line the mean value and standard deviation of the measured time periods.

Line topology is typically used in CNC systems. The number of nodes is small (usually less than 10) and they are located close to each other. Most devices have integrated two-port hubs dedicated to Ethernet Powerlink communication. Delays introduced by the drives are usually much smaller than

the overall communication cycle duration. Furthermore short connections between drives mean smaller EMI influence. Therefore line topology is used in the experimental station.

The experiment was focused on measuring EPL cycle jitter. Cycle jitter is the difference between EPL cycle periods marked by the Start of Cycle frames. SOC frames are synchronization frames that trigger the passing of a commanded position value to the servo loop. Therefore it is important for the period between consecutive SOC frames to be as stable as possible. SOC jitter is the most important parameter for this control system as it influences machining accuracy.

Two Baldor Microflex e100 MFE230A003 servo drives were used in the experiment along with two BSM63N-250AF PMSM motors. Computational load introduced by EMC2 is a major factor influencing jitter during normal operation of the proposed CNC control system. Therefore communication tests were performed while running a large g-code machining program in EMC2. Time periods between 20,000,000 consecutive SOC frames were measured for each PC controller by the EPL network analyzer. Communication cycle period was set as 1ms. Experimental results are presented in table 3, and in figure 10.

Maximum communication cycle jitter for all tested computer platforms does not exceed 12µs and its standard deviation is below 1µs. All mean values of the SOC period are slightly different than the demanded value of 1ms. This is caused by limited resolution of the hardware APIC timers, which trigger the beginning of the EPL cycle. A higher APIC operating frequency enables better timer resolution and less jitter. These small differences in cycle time do not affect the proper operation of the CNC controller as the trajectory planner uses real cycle value read from the hardware timer for its computations instead of the ideal one.

Maximum jitter and standard deviation are different for each computer. This is caused by different chipset architectures and their operating frequencies. Computer 2 which has the lowest jitter uses a high-end X48 chipset while computer 1 (worst jitter) utilizes an older 945G chipset which operates at lower speeds and generates more latency. Computer 3 utilizes a modern but lower-end H55 chipset therefore it exhibits higher jitter than the older but faster X48 chipset. In general, more modern and high-end chipsets operate at higher speeds and introduce less latency and jitter in real time operation.

At maximum machine federate of 30m/min (rotary motor maximum speed 3000rpm, ballscrew pitch 10mm/rev) displacement caused by 12µs jitter would be equal to 6µm. At normal working federate (5m/min) jitter induced displacement would equal to 1µm. For machines with required error tolerances of 0.01 – 0.05mm machining error introduced by communication jitter is insignificant. Applications of such machines include wood machining, plastics machining, machining of some metals, laser cutting, manipulators etc.

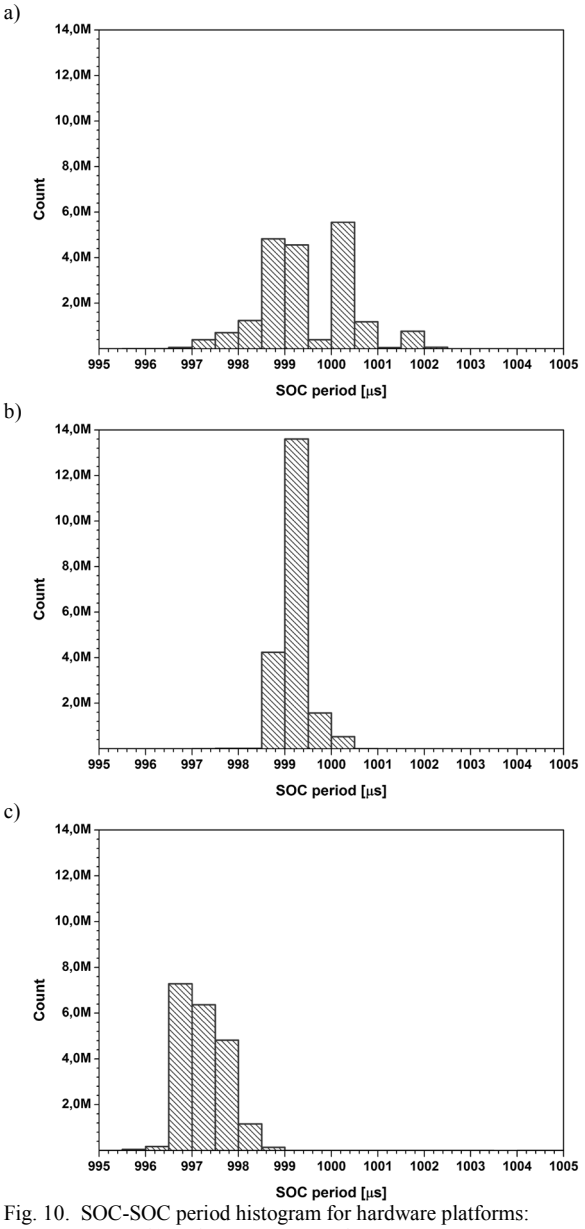

Fig. 10. SOC-SOC period histogram for hardware platforms: a) 1st computer, b) 2nd computer, c) 3rd computer.

TABLE III
SOC-SOC JITTER STATISTICS FOR 20 000 000 SAMPLES

Dataset	1st computer	2nd computer	3rd computer
mean val.[µs]	999.4565	999.2829	997.2869
max. range. [µs]	11.7918	9.9695	11.5113
std. dev. [µs]	0.9721	0.3433	0.4540

V. CONCLUSION

In this article the open PC-based CNC control system with Ethernet Powerlink communication protocol is presented. The main goal of this work was to create a low-cost, flexible, purely software CNC control system that can utilize various commercially available servo drives. Low cost is achieved by utilizing a general purpose PC with free open-source RTOS (Linux RTAI) and CNC control software (EMC2) without any dedicated hardware (e.g. FPGA or DSP expansion board). This configuration is highly flexible due to software-only implementation of the CNC controller and large computational

resources of the PC. New functionality can be easily added by writing new software modules.

Ethernet Powerlink protocol stack was implemented in the Linux RTAI real-time operating system. The stack was expanded with the CiA402 servo-drive communication profile and was integrated with the EMC2 CNC control software. Utilization of Ethernet Powerlink enables usage of different off-the-shelf servo drives produced by many manufacturers (e.g. Parker, Baldor) which adds to the system flexibility.

The results presented in this article prove that the presented system meets the requirements for a hard real-time CNC control system. Jitter measurements were performed while running a machining program in EMC2 to ensure stable operation under computational load. Maximum jitter of the Ethernet Powerlink communication cycle period is less than 12µs and its standard deviation is below 1µs for 3 different computer platforms. This result is highly satisfactory for many CNC motion control applications such as wood and plastics machining, machining of some metals, laser cutting, manipulation machines etc. In those applications machining error introduced by jitter of this magnitude is negligible (1µm) compared to required error tolerances (0.01 – 0.05mm).

Modifications to the EPL stack, Linux configuration (isolation of real-time tasks and real-time interrupts on separate processor cores) as well as BIOS and mainboard configuration (e.g. disabling unnecessary hardware, power saving options, System Management Interrupts) were required to ensure stable and precise operation

Future work will focus on enhancing the real-time performance by isolation of the Powerlink stack on a dedicated core separate from the CNC control application. An attempt to solve problems with interrupt sharing will be made by using message signaled interrupts (MSI) supported by modern PCI-E devices.

REFERENCES

[1] A. Malinowski, and Hao Yu, "Comparison of Embedded System Design for Industrial Applications," *IEEE Trans. Ind. Informat.*, Vol. 7, No. 2, May 2011, pp. 244-254.

[2] E. Monmasson, L. Idkhajine, M. N. Cirstea, I. Bahri, A. Tisan, and M. Wissem Naouar,, "FPGAs in Industrial Control Applications," *IEEE Trans. Ind. Informat.*, Vol. 7, No. 2, May 2011, pp. 224-243.

[3] Gu G.Y., Zhu L.M., Xiong Z.H., Ding H., "Design of a Distributed Multiaxis Motion Control System Using the IEEE-1394 Bus," *IEEE Trans. Ind. Electron.*, Vol. 57, No. 12, Dec. 2010, pp. 4209-4218.

[4] J. Jasperneite, J. Imtiaz, M. Schumacher, and K. Weber, "A Proposal for a Generic Real-Time Ethernet System," *IEEE Trans. Ind. Informat.*, Vol. 5, No. 2, May 2009, pp. 75-85.

[5] K. Kim, M. Sung, H. Jin, "Design and Implementation of a Delay-Guaranteed Motor Drive for Precision Motion Control," *IEEE Trans. Ind. Informat.*, Vol. 8, No. 2, May 2012, pp. 351-356.

[6] H. Carlsson, B. Svensson, F. Danielsson, B. Lennartson "Methods for Reliable Simulation-Based PLC Code Verification," *IEEE Trans. Ind. Informat.*, Vol. 8, No. 2, May 2012, pp. 267-278.

[7] T. Harmon, M. Schoeberl, R. Kirner, R. Klefstad, K. H. K. Kim, M. R. Lowry, "Fast, Interactive Worst-Case Execution Time Analysis With Back-Annotation," *IEEE Trans. Ind. Informat.*, Vol. 8, No. 2, May 2012, pp. 366-377.

[8] Chen Shuxin, and An Bin, "Time Performance Research on Field Bus Based CNC System," *2nd International Conference on Mechanical and Electronics Engineering (ICMEE)*, Vol. 2, Beijing, China, 1-3 Aug. 2010, pp. 56-59.

[9] Hu Chaobin, Li Wanli, and Xu Wuquan, "Study on the CNC system interpolation based on windows CE.NET and its real-time," *International Conference on Computer, Mechatronics, Control and Electronic Engineering (CMCE)*, Vol. 2, Tongji Univ., Shanghai, China, 24-26 Aug. 2010, pp. 110-112.

[10] D. Yashiro, and K. Ohnishi, "Performance Analysis of Bilateral Control System With Communication Bandwidth Constraint," ," *IEEE Trans. Ind. Electron.*, Vol. 58, No. 2, Feb. 2011, pp. 436-443.

[11] M. M. H. P. van den Heuvel, R. J. Bril, J. J. Lukkien, "Transparent Synchronization Protocols for Compositional Real-Time Systems," *IEEE Trans. Ind. Informat.*, Vol. 8, No. 2, May 2012, pp. 322-336.

[12] Tianrong Gao, Dong Yu, Dongfeng Vue, and Yi Hu, "Design and Implementation of Communication Platform in CNC System," *Mechatronics and Embedded Systems and Applications (MESA), 2010 IEEE/ASME International Conference,* Qingdao, ShanDong, 15-17 July 2010, pp. 355–360.

[13] A. Onat, T. Naskali, E. Parlakay, and O. Mutluer, "Control Over Imperfect Networks: Model-Based Predictive Networked Control Systems," *IEEE Trans. Ind. Electron.*, Vol. 58, No. 3, March 2011, pp. 905-913.

[14] P. Martí, A. Camacho, M. Velasco, and M. El Mongi Ben Gaid, "Runtime Allocation of Optional Control Jobs to a Set of CAN-Based Networked Control Systems," *IEEE Trans. Ind. Informat.*, Vol. 6, No. 4, November 2010, pp. 503-520.

[15] Á. Cuenca, J. Salt, A. Sala, and R. Pizá, "A Delay-Dependent Dual-Rate PID Controller Over an Ethernet Network," *IEEE Trans. Ind. Informat.*, Vol. 7, No. 1, February 2011, pp. 18-29.

[16] G. Cena, L. Seno, A. Valenzano, and C. Zunino, "On the Performance of IEEE 802.11e Wireless Infrastructures for Soft-Real-Time Industrial Applications," *IEEE Trans. Ind. Informat.*, Vol. 6, No. 3, August 2010, pp. 425-437.

[17] A. Mifdaoui, F. Frances, and C. Fraboul, "Performance Analysis of a Master/Slave Switched Ethernet for Military Embedded Applications," *IEEE Trans. Ind. Informat.*, Vol. 6, No. 4, November 2010, pp. 534-547.

[18] P. Ferrari, A. Flammini, S. Rinaldi, and E. Sisinni, "On the Seamless Interconnection of IEEE1588-Based Devices Using a PROFINET IO Infrastructure," *IEEE Trans. Ind. Informat.*, Vol. 6, No. 3, August 2010, pp. 381-392.

[19] Z. Hanzálek, P. Burget, and P. Šůcha, "Profinet IO IRT Message Scheduling With Temporal Constraints," *IEEE Trans. Ind. Informat.*, Vol. 6, No. 3, August 2010, pp. 369-380.

[20] *Real-time Ethernet SERCOS III*, IEC Std. 62410, 2005.

[21] *Industrial communication networks – Profiles – Part 2: Additional fieldbus profiles for real-time networks based on ISO/IEC 8802-3,* IEC Std. 61784-2, 2007.

[22] L. Dozio, and P. Mantegazza, "Linux Real Time Application Interface (RTAI) in low cost high performance motion control", *Motion Control 2003, a conference of ANIPLA, Associazione Nazionale Italiana per l'Automazione (National Italian Association for Automation)*, Milano, Italy, 27-28 Mar. 2003.

[23] *EMC2 User Manual V2.4,* The EMC Team, 2011.

[24] *Adjustable speed electrical power drive systems – Part 7-201: Generic interface and use of profiles for power drive systems – Profile type 1 specification,* IEC Std. 61800-7-201, 2007.

[25] M. Cereia, I. C. Bertolotti, and S. Scanzio, "Performance of a Real-Time EtherCAT Master Under Linux," *IEEE Trans. Ind. Informat.*, Vol. 7, No. 4, November 2011, pp. 679- 687.

[26] G. Cena, I. C. Bertolotti, S. Scanzio, A. Valenzano, and C. Zunino, "Evaluation of EtherCAT Distributed Clock Performance," *IEEE Trans. Ind. Informat.*, Vol. 8, No. 1, February 2012, pp. 20-29.

[27] *Numerical control of machines - Program format and definition of address words - Part 1 : Data format for positioning, line motion and contouring control systems,* ISO Std. 6983-1, 1982.

[28] *IEEE Standard for Information technology— Telecommunications and information exchange between systems— Local and metropolitan area networks— Specific requirements Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications,* IEEE Std. 802.3-2008, 2008.

[29] *Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model,* IEC Std. 7498-1, 1996.

[30] *Adjustable speed electrical power drive systems – Part 7-301: Generic interface and use of profiles for power drive systems – Mapping of profile type 1 to network technologies,* IEC Std. 61800-7-301, 2007.

[31] *Introduction into openPOWERLINK Software Manual*, SYS TEC electronic GmbH, 2008.