Recent Innovations in Mechatronics (RIiM) Vol. 2. (2015). No. 1-2.
DOI: 10.17667/riim.2015.1-2/6.

Industrial robotics for ERP controlled smart factories

Ferenc Tajti
MTA-ELTE Comparative Ethology Research Group (MTA: 01 031), Dpt of Mechatronics Optics and Engineering Informatics, Faculty of Mechanical Engineering
Budapest University of Technology and Economics
Budapest, Hungary
tajti@mogi.bme.hu

Géza Szayer
Department of Mechatronics Optics and Engineering Informatics, Faculty of Mechanical Engineering, Budapest University of Technology and Economics
Budapest, Hungary

Bence Kovács
Department of Mechatronics Optics and Engineering Informatics, Faculty of Mechanical Engineering, Budapest University of Technology and Economics
Budapest, Hungary

Péter Korondi
Department of Mechatronics Optics and Engineering Informatics, Faculty of Mechanical Engineering, Budapest University of Technology and Economics
3MTA- BME Control Engineering Research Group
Budapest, Hungary

***Abstract*—At product manufacturing the time-to-market factor, the profitability and the delivered value define the success of an enterprise. The increasing number of modules in Enterprise Resource Planning (ERP) programs is a facing problem, when there is a margin between the manufacturing cells and the ERP. Nowadays, the connection between the industrial machines and the ERP is an important requirement especially at automated warehouses and smart factories. Other concerns at manufacturing are the maintenance schedules of the machines, and flexible and easy reconfiguration of the production lines or the production cells. Information technology provides solutions and software environments to implement complex production supervisor ERPs at smart factories. At a production line or an automated warehouse several technical parameters and information can influence the planning of the resources at the enterprise, like maintenance, machine error, stockpile, product ID, defective product ratios, etc. When there is machine maintenance, the company needs to order the service parts, as well as schedule the service time and the stop of the production line. In case of a machine error, the system can estimate the length of the service time from error messages, and reorganize orders, transportation, or even maintenance schedules of other machines. Our plug and play type robot and industrial automation controller project gives a solution for these hardware demanding needs.**

Keywords— Intelligent systems; Robot control; Automation;

I. INTRODUCTION

Hardware solutions can be easily configured with any industrial machine or automation cell, and developed to LinuxCNC, which is an open source Linux software environment. The core of the system is a PCI based motion control card with several IOs and further RS485 and CAN based modules. According to the open source structure, any information of the system (like maintenance, machine error, stockpile, or product ID) can be monitored, saved and transmitted via telnet or other PC compatible communications. In this paper we present the developed PCI driver and the hardware environment around LinuxCNC. At automated warehouses with product and product part IDs the error rate and the cause of the problems can be investigated, and the stockpiles can always be updated and compared to the expected values [1, 2]. The ERP of a smart factory requires special machine information. Even in the same factory and the same machine the type of the information depends on the application [3]. For example the tool abrasion, the tool amortization and tool change can be also estimated and scheduled. When an industrial robot is used for painting, the required information is the amount of paint, but at a selection scenario it can be the color of the parts. At a production line the messages and the errors have different priorities. The amount of the red and green colorized parts is not as important as an error message. A robot servo fault error will stop the whole machine, and in most of the cases the whole production line. If a part is broken, that also generates error messages, but it influences only the error rate, and it warns the system to check the previous process or the supplier. At commercial industrial robots the access to information of a machine is limited, especially at PLC based production cells [4]. It is usually enough for normal system requirements, but not for special needs [5-8]. At the open source automation control system different modules can be installed, implemented or developed like at the ERPs [9, 10]. The development of LinuxCNC was started from previous projects in the United States Patent and Trademark Office. According to the open

source and modular software environment we could implement a PCI driver, that monitors all of the variables in the embedded system and at the RS485 and CAN modules as well. The PCI motion control card is an FPGA and PCI-bridge based ASIC (Application Specified Integrated Circuit) for interfacing with the most possible system elements that can be found in an automated robotic cell.

II. Software

A. Interface between ERP solutions and manufacturing

As the PCI card can be installed to any general purpose PC or industrial PC solutions as well, it can be connected to ERP servers via all conventional protocols, like Ethernet or wireless network. For a ready to use solution, the LinuxCNC kernel can communicate on Telnet interface via Ethernet of WiFi [11-13]. (See Figure 1).

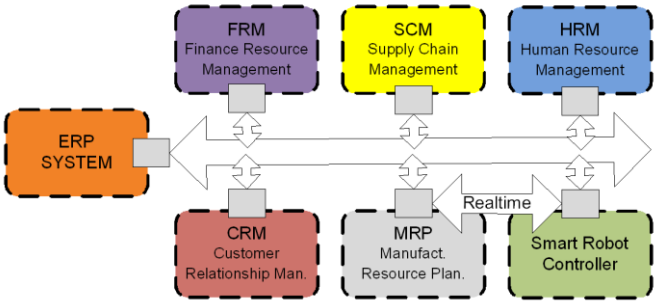

Figure 1 The block diagram of the ERP system with our robot controller concept

B. System description

In the software engineering sense the core of the system is the LinuxCNC (the Enhanced Machine Controller), an open-source software environment for computer control of industrial machines like milling machines, robots and lathes. The user can integrate self-developed software modules into the LinuxCNC, like a PCI card driver, direct, inverse kinematics, and communication frames via any PC-based communication interfaces and any other software modules. For example during a machine control with CAN bus and RS485, the PCI card handles the OSI L1-L3 layers of machine specific CAN and RS485 communication, while the LinuxCNC driver handles PCI communication and all the higher OSI Layers of machine interfaces.

C. LinuxCNC

As mentioned in the introduction, the development of LinuxCNC was started from previous projects in the United States Patent and Trademark Office. Source code is hosted by open source community using GIT version control system. The open source community motivates, and provides support for developers to add thousands of lines of new code and many new modules to LinuxCNC every month. LinuxCNC integrators may configure which modules to use and how to connect them. After a software development period is closed, all modifications are approved and tested, and the active and supportive community publishes the next main version on the official website with the additional functions, modules and documentation.

LinuxCNC software modules can be set up and configured based on the Hardware Abstraction Layer concept, like many operating systems. In this model, all software modules have well-defined inputs, outputs and functions. These software modules are compiled, but not connected in compile time. LinuxCNC integrators define connections and configure module parameters in HAL files. HAL files are loaded every time the program starts, so integrators do not need to compile software. The LinuxCNC has different software modules like kinematics, HAL files, user interfaces, scopes, ladder diagrams, etc. These software parts can be added and connected with each other. For example the following HAL software modules will be configured or implemented in the case of an industrial CNC machine with automatic work piece handling: kinematics, touch screen optimized user friendly GUI (see Figure 2), computer vision system and the ladder diagram for some PLC functions. This open and highly modular concept enables LinuxCNC to control many different physical systems [14]. and LinuxCNC can be controlled on higher level by ERP.

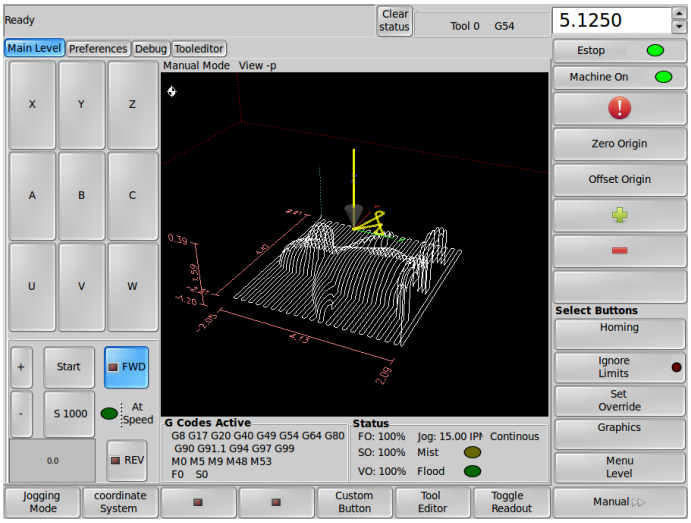

Figure 2 The touch-screen-optimized GUI of the Linux CNC interfaces.

D. Custom PCI driver

LinuxCNC runs on Real-Time Linux kernel (Linux RTAI). A new kernel module was developed to control custom PCI cards, as well as to interface custom robots and other industrial machines to LinuxCNC software environment. It remains as a challenge of the software and hardware specification to support new machine interfaces, like CAN or Ethernet based field buses, but it supports conventional robotics and automation interfaces as well, like incremental pulse motion reference signals.

In case of incremental position reference, the step signals and ramp functions at the system are generated by the PCI

card. The most interesting implementation is the step generation, but the methods below are used at least partly at the analogue, Ethernet based, and other CAN based references and control methods, like CAN-Open or Device-Net. (See Figure 3)

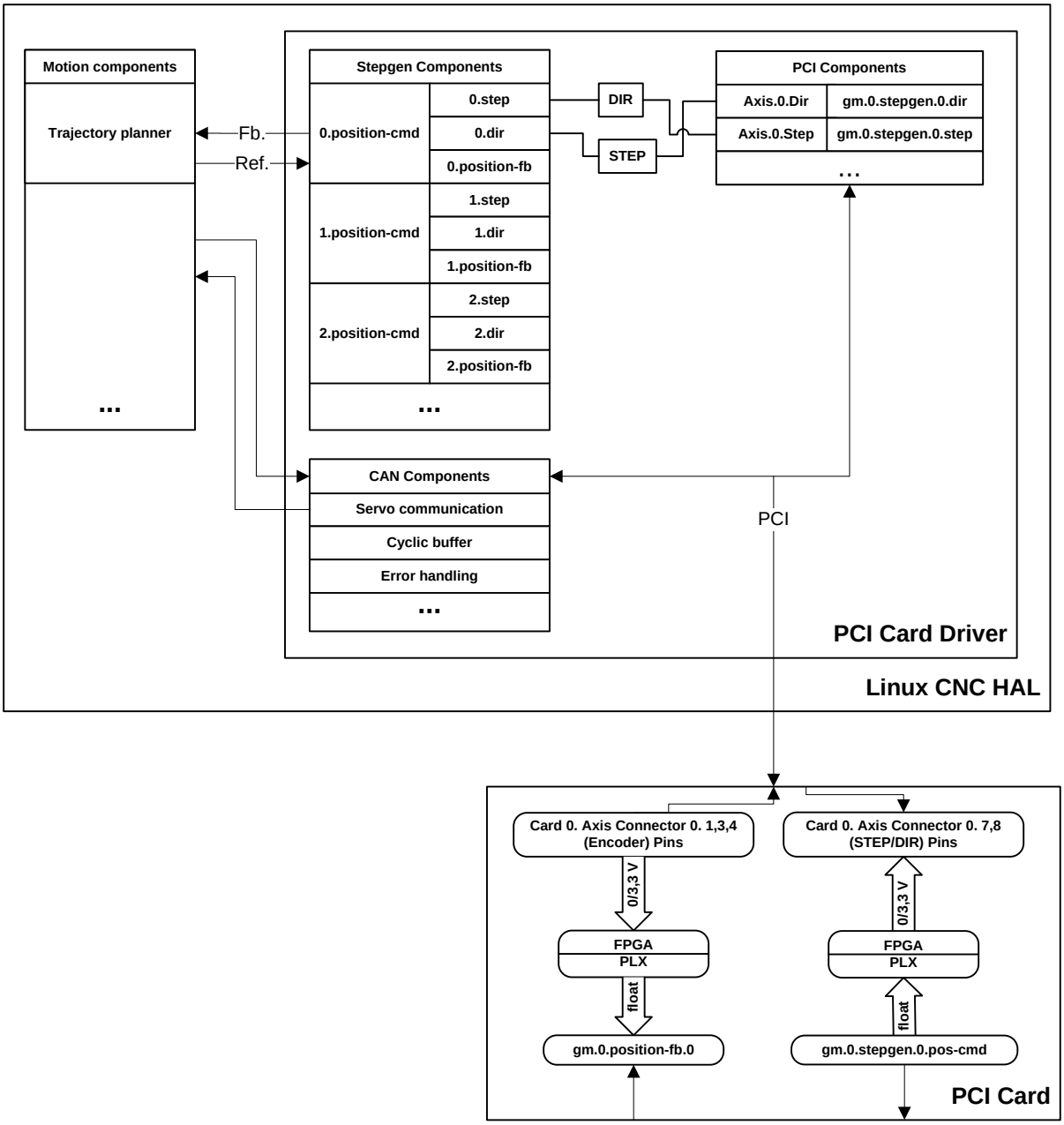

Figure 3 The block diagram of the HAL layer

The input of the step generator (in line with the constant variables) is the position reference, and the outputs are the step and the dir signals. The basic principle of this speed control based interpolation is that the interpolator runs at a fixed period interrupt, and calculates the required speed for the motor in every time step, following the required motion profile. To start and stop the motor in a smooth way without steady state position error or position overshoot, control of the acceleration and deceleration is needed. The relation between (angular) acceleration (see equation 1), speed (see equation 2) and position (see equation 3) is the derivation. (See figure 4).

$$\ddot{\varphi} = \xi \tag{1}$$

$$\dot{\varphi} = \omega = \xi\, t \left(+ C_1\right) \tag{2}$$

$$\varphi = \frac{1}{2}\, \xi\, t^2 \left(+ C_1\, t + C_2\right) \tag{3}$$

The position error (see equation 4) is calculated by the subtraction of the actual position from the reference position, or "position command".

$$\varphi_{error} = \varphi_{actual} - \varphi_{reference} \tag{4}$$

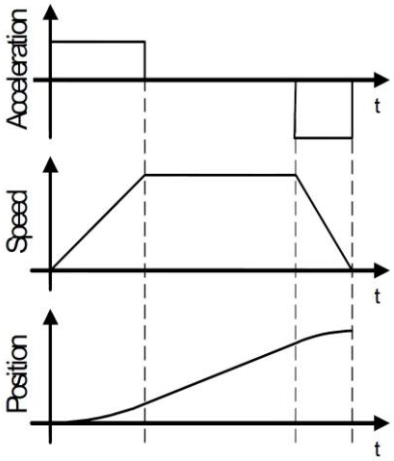

Figure 4 Position, speed and acceleration time functions

Initially, the speed has to be calculated from the position error constrained by the given maximum deceleration that drives the motor exactly to the target position. In other words, the speed has to be limited for preventing position overshoot. For this, we have to express the speed from the position and speed motion equations (see equation 5).

$$\omega = \xi \sqrt{\frac{2\varphi_{error}}{\xi}} = \sqrt{2\,\varphi_{error} \cdot \xi} \tag{5}$$

The second constraint is the calculated speed which doesn't cause position overshoot, and has to be limited to a maximum value that the motor can follow. The last constraint that has to be considered is the maximum acceleration that can decrease the actual speed output of the interpolator. (See Figure 5) The requested acceleration can be calculated by numeric derivation, by subtracting the last calculated speed from the actually calculated one. Then it can be saturated by simple branches. If the required acceleration is higher, the new speed output will be equal to the last speed output incremented by the maximum acceleration.

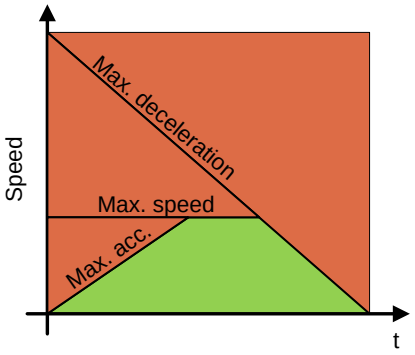

Figure 5 The ramp function

The calculated speed in step/s has to be converted to timer count period for evaluating the step generation (see equation 6).

$$Period = \left| \frac{\left[timer\,frequency \right]}{\omega} \right| \tag{6}$$

The period can be calculated only if the speed output is non-zero. Another variable has to indicate if the motor has to be stopped. Hence the period does not include the direction information; the direction has to be stored in another logic variable, the value of which depends on the sign of the speed.
The timer compare registers value has to be calculated for the next step in each interrupt. The period variable is given in timer counts. So the next compare value can be calculated by adding the period to the last compare value, but the register and the timer overflow have to be handled. If the period exceeds the timer top value, and overflow counter has to be set depending on the number of overflows. There is no step generation in every interrupt request in this case, only when the overflow counter is zero again. The software is composed from a fixed and a variable period timer interrupts. The fixed step interrupt is practically executed in between 1~10 ms. The position error, the speed, the timer period and the direction are calculated in order in this interrupt. First, the timer compare registers next value is calculated in the variable period (step generation) timer interrupt. Then one step should be generated, if the Run flag is active. After a step, the actual position has to be updated for giving position feedback for the interpolator. The speed is calculated in the fixed period interrupt, and during deceleration, the zero speed should be given when the position error is zero. But in most cases, more than one step is generated in the variable period timer interrupt with the last speed command; hence position overshoot and oscillation occur. (See Figure 6)

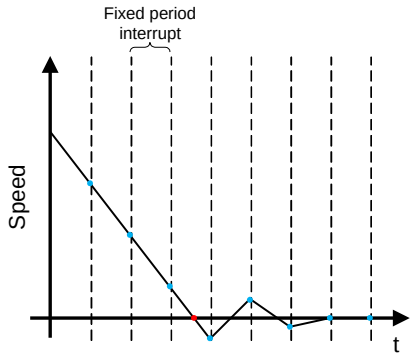

Figure 6 The overshoot function

To avoid this situation, the conditions of the final stop have to be handled in the variable period timer interrupt in case of all steps. (See Figure 7) The conditions are as follows: If the position error is zero, and the speed is under a low speed threshold, then do not generate any more steps (red point in Figure 7). Practically, a low speed flag should be set or cleared in the fixed period interrupt, for improving the execution efficiency (flsf (Speed)). The comparison of the signed 16 bit values is done in the fixed period interrupt. And only the low speed flag is checked for every step in the variable period (high speed) interrupt.

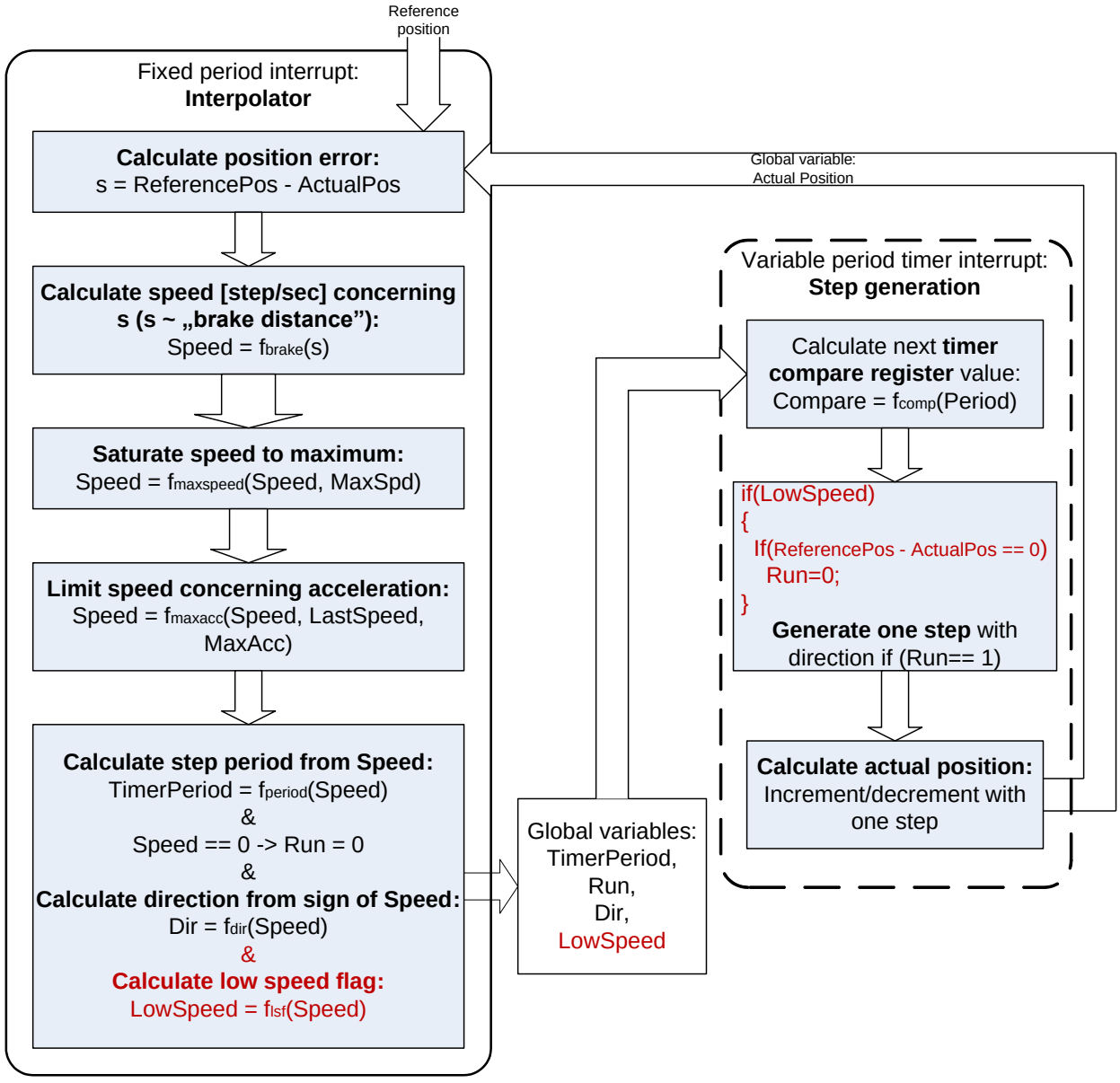

Figure 7 The detailed block diagram of the step generation

III. HARDWARE

A. System description

The system can be configured with different servo applications and PLC functions. The center of the controller is the PCI based motion control card. The main goal of the development is to follow the newest paradigms in the view of robot control theory, what means that the system should be open source and plug and play. At the integration of a manufacturing cell at a production line, different kind of signs and information of the robot may need to be observed at a smart factory system [15]. At a normal industrial robot control system just the typical information can be monitored, like the position and the orientation of the robot, the joint coordinates and the number of the working cycles etc. At an open source and plug and play type automation controller all of the parameters, variables and signs can be monitored during the manufacturing or the storage process. The PCI card can be used in a normal PC with a real-time Linux OS. Teach pendant with VGA, touch interface with USB, Ethernet, camera vision system and other PC based or custom informatics devices can be used with the machine in the view of information science and ERP-s.

In the view of robot control theory the card has a PLX controller with PCI based communication, and it is connected to a Xilinxs Spartan III FPGA via a 32 bit parallel bus. (See Figure 8) The FPGA and the PCI driver handle the connection between the hardware and the software layer. Most of the robot controllers cannot handle PLC functions and cannot be connected to a network as flexible as a normal real-time Linux based PC. The analog and digital RS485 IOs can be controlled by a normal C code or the ladder diagram editor of the software environment like at normal PLC-s. With the different isolator, differential line driver, IO, and breakout modules at a production line most of the elements can be controlled including the production specific special machines like a five axis milling machine, which produces tooth implants for dentists. In these cases the machine is designed by mechanical engineers and it cannot be integrated into a manufacturing cell without a controller. The system can be connected to these specialized machines without any long time demanding software and electrical engineering development [16]. The CAN based, analog or incremental servo amplifiers of the machine can be connected to the PCI card directly or via opto-isolated and differential driver modules. Six axes can be interpolated with one card and the LinuxCNC software environment can handle two cards and can interpolate up to nine axes. The RS485 bus can handle 16 modules (relay output, opto-isolated input, ADC and DAC, Teach Pendant) with one card. (See Figure 9)

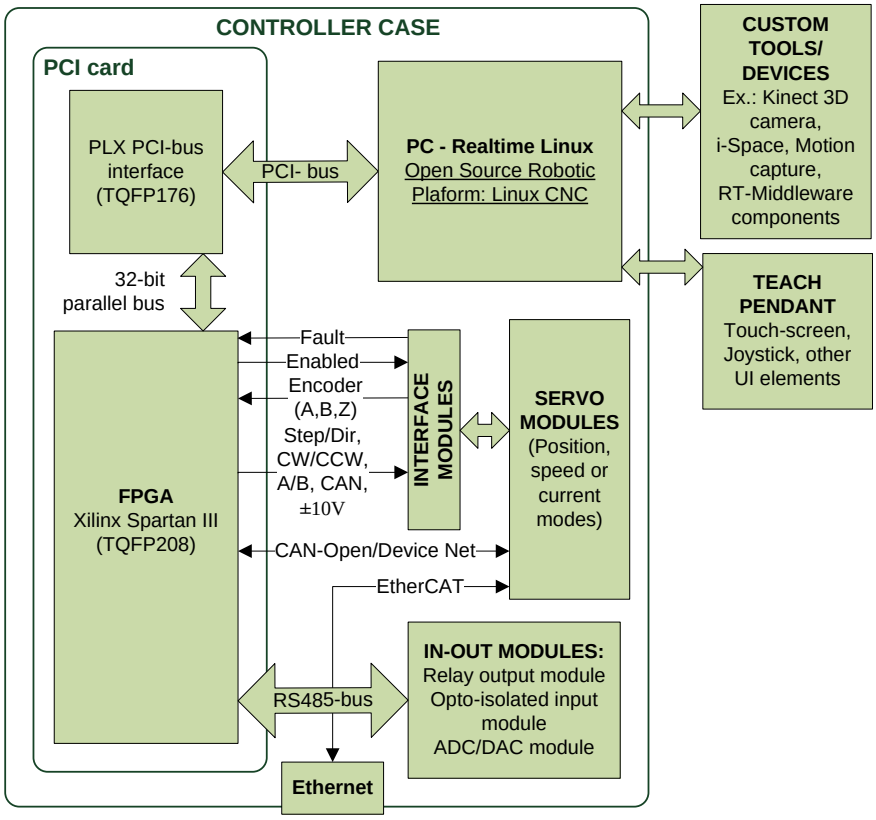

Figure 8 The detailed block diagram of the system

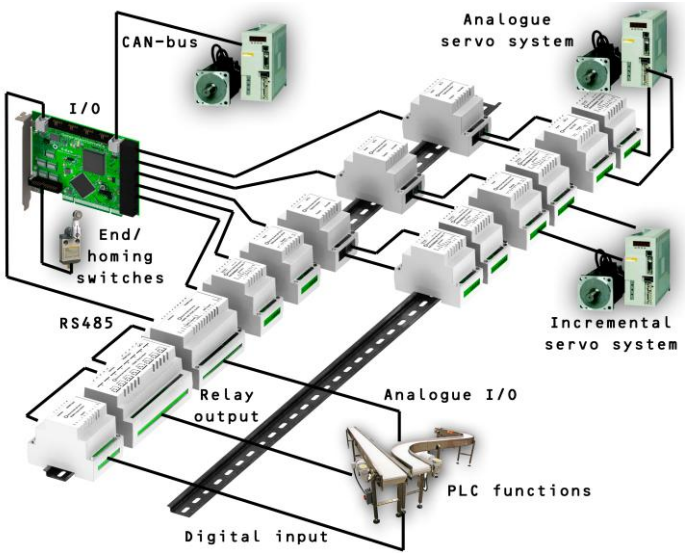

Figure 9 The block diagram of a sample robot controller

B. PCI card

As it is detailed below the card has a PLX-Xilinx embedded system and it can be implemented in a production line without hardware engineering development. This advantage comes from the modularity, the fact that the software environment is open source, and the peripheries of the PCI card (see Figure 10). The main hardware elements for the peripheries are the parts of the card, like the isolated CAN, the RS485 transceivers, the opto-couplers of the machine specific pins, the monitoring LEDs, six axes connectors, and four connectors for general purpose IO pins. The RS485, CAN, incremental servo, axis DAC and other modules can be connected directly to the card.

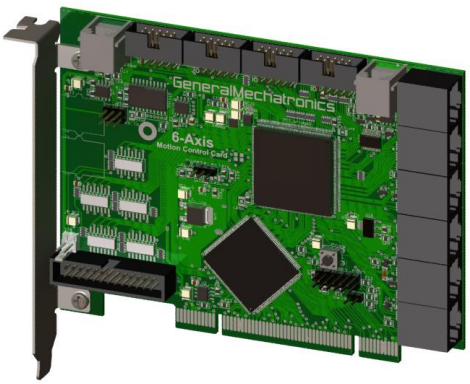

Figure 10 The PCI card

Some of the registers can be written to (like step variables) and some of the registers can be read (like encoder variables). At discrete control (series of coordinates) for the robot we only calculate the next point (position, joint coordinate), and the path is not prescribed between the current and the next position. At continuous control the interpolator of the system makes continuous calculations to get path points frequently and create a smooth motion between the start and the end position and orientation. The software environment writes the joint coordinates of these positions and orientations to the corresponding registers during the motion. The motion control card and the PCI driver generate the physical form of the data like step signs for the incremental servo amplifiers. The position or speed controllers are calculated on the servo amplifiers. In case of an old analogue system the position and speed control methods can be calculated on the PCI card. This method can handle decentralized motion controls. In case of a centralized motion control we can implement the inverse dynamical equations of the machine so torque (current) references can be given to the servo amplifiers. Some of the control methods and parameters are calculated on the PCI card even for decentralized motion control. For example at a step signal generation we can define the maximal acceleration, the maximal speed and the maximal deceleration, what describes the definition of the velocity-time function of an axis. The length of the steps, the direction delay, the length of the step spaces and the step/unit can also be adjusted. The ramp function generates the step signals between two discrete time steps during the motion control.

C. Servo applications

The servo motors of the industrial robots are usually PMSM or BLDCs and at older robots are brushed DC motors. The servos amplifiers are usually sold in pair with the motors, which can get position, velocity, or torque references via incremental, CAN based, or analog signal references.
The following typical example connections can be easily made between the PCI card and the servo amplifiers with the developed modules of the system:

Analogue system with encoder feedback

- Incremental digital system with encoder feedback and differential output

- Incremental digital system with encoder feedback and TTL output (see Figure 11)
- Incremental digital system with differential output
- Incremental digital system with TTL output
- Absolute digital (CAN based: CAN-Open) system (see Figure 12)
- Absolute digital (CAN based: CAN-Open) system with conventional (A/B/I) encoder feedback

Each of the servo connections has a detailed block diagram. In each application the signals between the servo amplifiers and the PCI card are isolated, and position feedback can be made to monitor the tracking errors of the TCP or the joints in the LinuxCNC. At the implementation of the system, during the tuning phase of the control methods (usually PI, PD, PID, or PID with anti windup) at the servo amplifiers, the step response can be monitored with the scope module of the LinuxCNC. The hardware parameters like the encoder CPR, the step/radian or step/mm, etc. can be defined in the HAL (Hardware Abstraction Layer) files at the configuration of the system. (See below in section 3.)

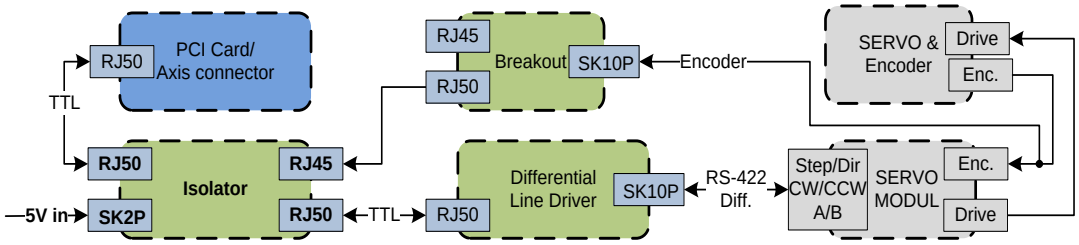

Figure 11 The block diagram of the differential axis control example

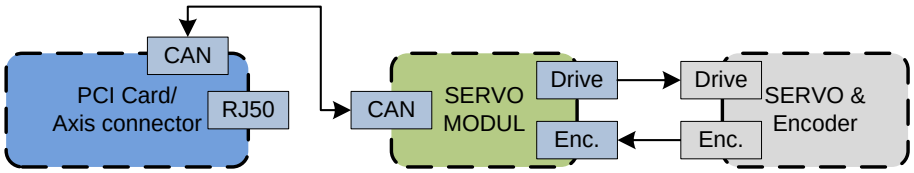

Figure 12 The block diagram of a CAN-based axis control example

D. RS485 modules

The PLC functions, tool magazines, conveyor belts, capacitive or infrared sensors can be handled by the developed RS485 modules of the system. In most of the machines one PCI card and one real-time Linux based PC are enough for the control of the servo modules and the peripheries. All of the modules are isolated, and have double connectors to create from module to module wires, with the PCI card at the start of the bus, and with end resistors at the end of the bus (See Figure 13). The system's IO capability is flexible: One input module can handle 8 isolated inputs, and a relay output module has 8 outputs. The analogue module has 4 DAC and 8 ADC channels. One card can handle 16 modules, so a complex CNC can be controlled even if it requires 50 relays to handle all of the peripheries. The variables of the RS485 modules (the values of the inputs and outputs) can be connected at the HAL files with any parts of the LinuxCNC software, for example with a button at the touch interface or the telnet module for further data handling.

The communication is a 9 bit RS485, where every module has a 4 bit address and a 4 bit command and the 9th bit indicates if it is a data message or an address/command message. The end of the communication there is a checksum. At the open source PCI driver, custom RS495 (and CAN) based communication protocol can be also defined by the user. It allows for the use products from different distributors or even self designed modules to optimize the manufacturing cell in the engineering sense.

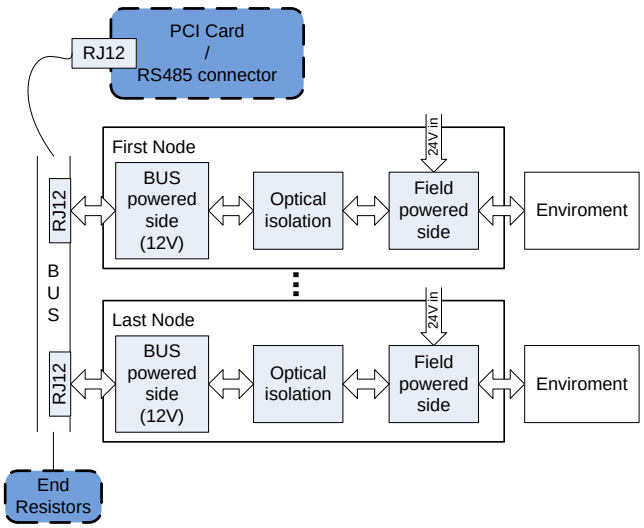

Figure 13 The block diagram of the RS485 modules

IV. Discussion

The development of this open source controller started in 2009, with the goal to create a traditional but easy to integrate, plug & play controller which can be customized for specific

needs. The experience of the last four years showed us an ongoing paradigm shift in the integration of industrial controllers into the ERP network of the smart factories. Due to the open source property, these controllers become a good platform for developing novel functions such as ERP integration.

V. CONCLUSION

In the past years, two of the first custom CNC controllers are integrated into a custom ERP system which was developed for automated dental milling cells. These controllers get their manufacturing information from a higher level management and resource planner system which handles the orders, work pieces, CAD/CAM files. And during manufacturing they send process and diagnostic feedback information for reporting and real-time monitoring.

Acknowledgment

This research was funded by the Hungarian Academy of Sciences (MTA 01 031). The authors wish to thank the support to the Hungarian Research Fund (OTKA K100951).

References

[1] Anke J., Kabitzsch K., "Cost-based Deployment Planning for Components in Smart Item Environments", in Emerging Technologies and Factory Automation, ISBN: 0-7803-9758-4, pp. 1238-1245, Prague, September 2006.

[2] Péter Tamás, Norbert Szakály, "Decision help system supported data-mining method", in Biomechanica Hungarics, ISSN: 2060-0305, Paper A0023, 2013.

[3] Yi Wu, Fengping Wu, "Researches on SAP-CRM's application in cigarette manufacturing factory", in Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), ISBN: 978-1-4577-0535-9, pp. 4762-4765, August, 2011.

[4] Mahir Dursun, Semih Özden, "PC-based data acquisition system for PLC-controlled linear switched reluctance motor", in Tuskish Journal of Electrical Engineering & Computer Sciences, 2013/21, doi:10.3906/elk-1105-24, pp. 71-80, January, 2013.

[5] Wolf B., Herzig P., Behrens I., Majumdar A., "Data stream processing in factory automation", in Emerging Technologies and Factory Automation (ETFA), ISBN: 978-1-4244-6848-5, pp. 1-8, Bilbao, 2010.

[6] Savio D., Karnoukskos S., Morieira Sa de Souza L., Trifa V., "Reactive business processes for factory automation", in Industrial Informatics Conference, ISBN: 978-1-4244-3759-7, pp. 620-625, Cardiff, Wales, June, 2009.

[7] Rashid M.A., Riaz Z., Turan E., Haskilic V., Sunje A., Khan N., "Smart factory: E-business perspective of enhanced ERP in aircraft manufacturing industry", in Technology Management for Emerging Technologies (PICMET), ISBN: 978-1-4673-2853-1, pp. 3262-3275, Vancouver, BC, August, 2012.

[8] van Putten, B.-J., Kuestner, M., Rosjat, M., "The Future Factory Initiative at SAP Research", in Emerging Technologies & Factory Automation, ISBN: 978-1-4244-2727-7, pp. 1-4, Mallorca, September, 2009.

[9] Ding Lijie, Tang Hao, Zhou Lei, "The Application of Peer to Peer SAP-based Q-Learning in Task Assignment to Multiple Robots", in Control Conference, ISBN: 978-7-81124-055-9, pp. 536-539, Hunan, June, 2007.

[10] Hameed B., Khan I., Durr F., Rothermel K., "An RFID based consistency management framework for production monitoring in a smart real-time factory", in Internet of Things, ISBN: 978-1-4244-7413-4, pp. 1-8, Tokyo, December 2010.

[11] Zubia J.G., Parra I.T., "WEB 2.0 control architecture for industrial robots", in Emerging Technologies and Factory Automation (ETFA), ISBN: 978-1-4244-6848-5, pp. 1-8, Bilbao, September, 2010.

[12] Alvares A.J., de Souza J.L.N., Teixeira E.L.S., Ferreira J.C.E., "A methodology for web-based manufacturing management and control", in Automation Science and Engineering, ISBN: 978-1-4244-2022-3, pp. 668-673, Arlington, VA, August, 2008.

[13] Behrouz Shahgholi Ghahfarokhi, Naser Movahedinia, "Contect gathering and management for centralized context-aware handover in heterogenous mobile networks", in Tuskish Journal of Electrical Engineering & Computer Sciences, 2012/6, doi:10.3906/elk-1101-1042, pp. 914-933, June, 2012.

[14] Erwinski, K. Paprocki, M. ; Grzesiak, L.M. ; Karwowski, K. ; Wawrzak, A. "Application of Ethernet Powerlink for Communication in a Linux RTAI Open CNC system" in IEEE Transaction on industrial electronics, ISSN: 0278-0046, pp. 628-636, February, 2013.

[15] Tyrin I., Vylegzhanin A., Kozhevnikov S., Kuznetsov O., Skobelev P., Kolbova E., Shepilov Y., "Multi-agent system "Smart Factory" for real-time workshop management: Results of design & implementation for Izhevsk Axion-Holding Factory", in Emerging Technologies & Factory Automation (ETFA), ISBN: 978-1-4673-4735-8, pp. 1-4, Krakow, September, 2012.

[16] Mesnil P.C., Ulmer C., Gomez L., "Web based communication between embedded systems and an ERP", in Industrial Informatics (INDIN), ISBN: 978-1-4244-3759-7, pp. 551-556, Cardiff, Wales, June, 2009.