ISSN 1330-3651(Print), ISSN 1848-6339 (Online)
UDC/UDK 004.451.2:621.914.3-52

LINUXCNC – THE ENHANCED MACHINE CONTROLLER: APPLICATION AND AN OVERVIEW

Tomislav Staroveški, Danko Brezak, Toma Udiljak

Subject review
This paper presents an application and a critical overview of the Enhanced Machine Controller (EMC2), a Linux based CNC open architecture control system (OAC) implemented in an industrial milling testbed platform. The development of such a machine tool was motivated by educational and research requirements, especially in the field of design and analysis of machining process monitoring and control algorithms. The EMC2 was analysed in view of compatible hardware components and software interfaces, configuration abilities and industrial applicability. Also, characteristics of the developed testbed together with the basic implementation details are depicted.

Keywords: *Computer Numerical Control (CNC), enhanced machine controller, LinuxCNC, open architecture controller, machine tool retrofitting, milling machine testbed*

LinuxCNC – Napredni sustav CNC upravljanja: primjena i kritički osvrt

Pregledni članak
Ovaj rad prikazuje osvrt na implementaciju upravljačkog sustava otvorene arhitekture, tzv. Enhanced Machine Controller-a (EMC2), koji je primijenjen na glodalici kao ispitnom postavu. Razvoj ispitnog postava motiviran je edukacijskim i istraživačkim potrebama, osobito u području razvoja i analize algoritama za nadzor i upravljanje procesima obrade odvajanjem. EMC2 sustav je razmatran s obzirom na kompatibilne hardverske komponente i mogućnosti prilagodbe specifičnostima obradnog stroja, kao i mogućnostima za industrijsku primjenu. U radu su također prikazane tehničke karakteristike ispitnog postava i upravljačkog sustava.

Ključne riječi: *numerički upravljani alatni strojevi (CNC), LinuxCNC, upravljački sustavi otvorene arhitekture, revitalizacija alatnih strojeva, ispitni postav za glodanje*

1 Introduction

In the past two decades, considerable effort has been put in the development of open control systems for machine tools. Machine tool manufacturers recognized these systems as a solution to the problem of how to define common concepts, develop basic technologies and produce basic components together in order to fulfill constant demands for higher machine tool functionality and flexibility, product quality and cost reduction [1]. According to the IEEE standards (1003.0), "an open system provides capabilities that enable properly implemented applications to run on a variety of platforms from multiple vendors, to interoperate with other system applications and present a consistent style of interaction with the user". This means that an Open Architecture Controller (OAC) has to be flexible in hardware and software for all control levels, i.e. it has to be standardized with respect to integration with other control systems and has to permit the integration of independent application program modules, control algorithms, sensors and computer software/hardware developed by different manufacturers [2]. With the ability of implementation and integration of customer-specific controls by means of open interfaces and configuration methods, the implementation of OAC is particularly important in the development of reconfigurable manufacturing systems [3] and in the development of advanced manufacturing systems with higher degree of autonomous operation accomplished by process monitoring and control modules.

The first OAC solution was proposed by the US National Institute of Standards and Technology (NIST) and was named Enhanced Machine Controller (EMC) [4]. It had all necessary features of an advanced numerical control system which can be implemented using an ordinary PC hardware running FreeBSD or Linux. After this first initiative, several other projects have been launched in Europe, the USA and Japan, among which the most important are [5]:

- OSACA (Open System Architecture for Controls within Automation System),
- OMAC (Open Modular Architecture Controllers),
- OSEC (Open System Environment for Controller),
- JOP (Japanese Open Promotion Group).

These projects were set up and supported by different machine tool makers, control and software vendors, system integrators, end users and academics. Besides the aforementioned, university research activities in hardware and software area of open architecture CNC systems have also resulted in different manufacturing systems presented in [6 ÷ 14]. Also, several commercial systems such as Delta Tau PMAC-NC, IBH PA 8000, Galil DMC 1000 and others have been developed [15].

However, despite all these efforts, applications of open CNC architecture systems are still substantially limited to the research and development area because of the liability and standardization issues [15]. Moreover, according to [5], some open control systems generally offer a possibility for modifications in the non-real-time environment in a fixed software topology. Authors of that survey paper also emphasized that they lack the necessary flexibility and are not based on vendor-neutral standards.

This paper presents an overview of the application of the second, more advanced version of the EMC software (EMC2) which was used as a CNC solution for a redesigned three-axis milling testbed machine. The machine was developed on the basis of technologically outdated industrial planning machine, which was retrofitted for the purpose of the project for intelligent monitoring of a machine tool main spindle. Its aim was to develop an industry-applicable advanced motor spindle unit with monitoring and diagnostic characteristics which could identify or predict motor spindle failures and tool breakage locally and remotely (WAN/LAN). Additionally, the development of this machine tool was also fostered by a need for an open-controlled platform which could be used in the analysis of different control algorithms and process monitoring techniques.

The EMC2 software, which runs on Linux-based operating systems (OS) with real-time extensions, was chosen because of its free open source modular structure, code maturity, high stability and quality performance. Considerable potential for commercial EMC2 implementation has been shown by several retrofitted/ developed prototype machines, including systems with complex kinematics. Currently, there are also several didactic EMC-based machines available on the market [16].

In this work, version V2.3 of the EMC2 software was chosen because it was the last and the most comprehensive version at the time. In the meantime, this version was replaced by newer, somewhat improved versions with fewer bugs and with several new features. Furthermore, the name of the software was also changed into the project name – LinuxCNC.

Characteristics of machine tool elements and their design, of the control system hardware and of the EMC2 system, together with implementation details and remarks, are given in the following sections.

2 EMC Overview

The first version of EMC was originally developed by the Intelligent Systems Division at the NIST. The work resulting from efforts to produce a motion control package as a test platform for concepts and standards remained in the public domain and the EMC quickly received attention among open source community which continued to maintain the software. In 2003, the EMC2 version was introduced comprising several important features that simplified, organized and extended the original version [17]. The most significant advances of the EMC2 were the introduction of Hardware Abstraction Layer (HAL) module, which greatly simplifies the interface to the control hardware. Major code optimizations have also been done as well as an extended support for implementation in a variety of different machines with complex kinematic structures.

Real-time extensions to Linux kernel are required for normal EMC2 operation, although it is possible to run the EMC2 in a non-real-time environment for simulation purposes without interfacing the actual hardware. Support currently exists for Linux kernel versions 2.4 and 2.6 with real-time extensions applied by either RTAI [18] or RT-Linux [19] patches. Experimental versions of EMC2 are also built for 64-bit kernels.

EMC2 consists of four main components (Fig. 1):

- Motion controller (EMCMOT),
- Discrete I/O controller (EMCIO),
- Task coordinating module (EMCTASK),
- Text-based and graphical user interfaces (GUI).

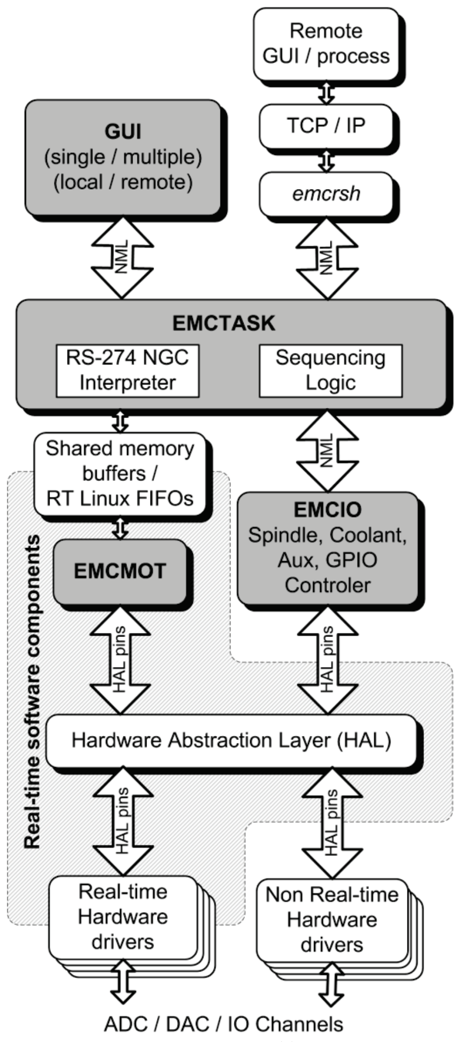
Remote GUI / process
TCP / IP
emcrsh
GUI (single / multiple) (local / remote)
NML
NML
EMCTASK
RS-274 NGC Interpreter
Sequencing Logic
Shared memory buffers / RT Linux FIFOs
NML
EMCIO Spindle, Coolant, Aux, GPIO Controler
EMCMOT
Real-time software components
HAL pins
HAL pins
Hardware Abstraction Layer (HAL)
HAL pins
HAL pins
Real-time Hardware drivers
Non Real-time Hardware drivers
ADC / DAC / IO Channels

Figure 1 EMC2 architecture

Among the four modules of EMC2, only EMCMOT is a real-time module. The communications between non-real-time modules are implemented by Neutral Messaging Language (NML) channels [20], and the communications between the real-time module (EMCMOT) and the non-real-time module (EMCTASK) is implemented either by shared memory or RT-Linux FIFO mechanisms [21].

2.1 Hardware Abstraction Layer (HAL)

As mentioned previously, in order to further expand EMC reconfiguration characteristics, Hardware Abstraction Layer (HAL), which is hierarchically placed between the EMCMOT/EMCIO modules and hardware drivers (Fig. 1), is introduced. HAL is a dynamic end-user reconfigurable mechanism which allows real-time data transfer between the EMC2 modules and from the EMC2 to the control hardware or other lower-level software modules. It also provides a framework for the development of hardware drivers and software modules that can be implemented in the EMC2 and executed in real time [22].

Modules can be dynamically loaded, unloaded and interconnected within the HAL space while the EMC2 is running. Each instantiated module is visible within the HAL space as a black box consisting of corresponding pins that serve as inputs and/or outputs. Connections between the HAL modules are established by creating corresponding HAL signals which must be of the same type as the module pins they interconnect (float, signed/unsigned integer or Boolean).

The development of customized real-time HAL modules is supported by two utilities: *Comp* and *Classicladder*. *Comp* is used as a versatile utility intended for facilitating the development, compilation and installation of more complex user modules written in the C language. *Classicladder* is used for the ladder logic-based module development. Support libraries are also provided for the Python language, which can be used for the coding of modules not intended for the execution in the real-time environment.

2.2 Motion controller (EMCMOT)

EMCMOT executes cyclically in real time and performs trajectory planning, direct and inverse kinematic calculations and computation of a desired output to the i^{th} motor control subsystem. This process includes the sampling of controlled axis positions, computation of the next trajectory point and interpolation between these trajectory points. Programmable software limits are also supported, as well as interfaces to hardware limit and home switches. Fig. 2 shows EMCMOT structure in greater detail.

Supported modes of operation are individual axis jogging (continuous, incremental, absolute), queued blended moves for linear and generalized circular motion, as well as programmable forward and inverse kinematics. Complex kinematics for robots can be coded in C language, according to a prescribed function interface and linked in to replace the default 3-axis Cartesian machine kinematics routines.

Parameters, such as number and type of axes (linear or rotary), scale factors between feedback devices (encoder counts), maximal velocity and acceleration values, axis units and trajectory planning cycle times are obtained from configuration file during system initialization phase.

Shared memory or FIFO mechanisms are used to receive commands or send status, error, or logging information to user space modules. NML is not used directly by the motion controller since NML requires C++ and the motion controller coding was limited to C in order to provide generic structure and maximize portability to other real-time operating systems.

EMCMOT interacts with subordinate real-time modules, such as PID compensation algorithms and other hardware drivers using HAL signals. Using the provided application-programming interface (API) for EMCMOT, a specific hardware support can be integrated into the EMC2 in the form of HAL modules, without modifying any of the core control codes.

For servo systems, the output is typically based on a PID compensation algorithm, provided in the form of separate HAL module. PID controller structures include zero, first, and second order feed-forward gains as well as maximum following error output. For stepper systems, the calculations typically run open-loop, and step generator, also written as HAL module, is used to send pulses to the stepper motor drives. This modular structure allows implementation of different compensation algorithms.

2.3 Discrete I/O controller (EMCIO)

The EMCIO module handles all I/O functions, which are not directly related to the actual motion of machine axis. It is implemented as an I/O controller consisting of a hierarchy of subordinate controllers for the main spindle, automatic tool change, the coolant, auxiliary functions (E-STOP chain, lubrication, etc.) and other user-defined subsystems.

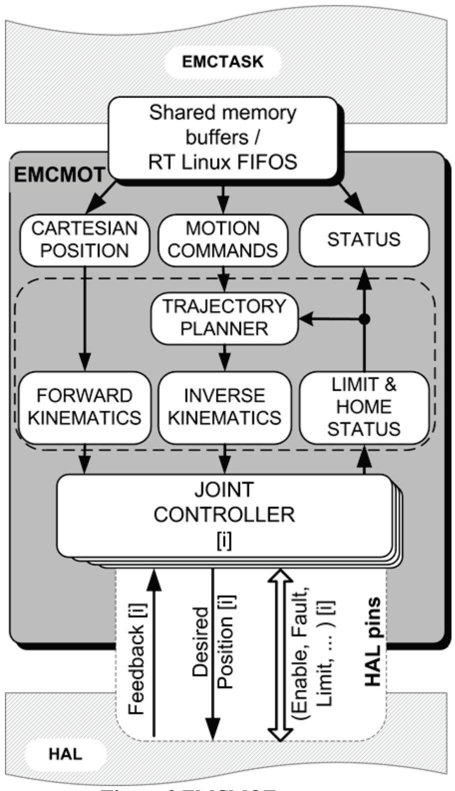

Figure 2 EMCMOT structure

Since discrete I/O controller design is typically highly machine-specific, customization in general is not intended using single configuration file used to configure the more generic EMCMOT. Instead, configuration is written in the form of single or multiple HAL files, each describing required application I/O interface or particular subsystem. These configuration files contain declarations of various HAL modules to be used, as well as HAL signals, by means of which module interconnections are described. Module interconnections are defined using signals of same type as corresponding module pins (binary, float, signed or unsigned integer). References to HAL configuration files are then included in main configuration file to be read during system startup.

Most of previously mentioned subsystems are already preconfigured in EMC2, but can easily be adapted or left unutilized, leaving only a subset that fits application needs. In addition, support for writing custom user-defined subsystems is provided either by means of integrated Programmable Logic Controller module (PLC) [23], or by utilizing specialized tool for writing HAL modules (*Comp*). PLC is configured during system setup, using ladder logic diagrams within specialized GUI (*ClassicLadder*), while *Comp* is used to build, compile and install HAL modules from source code.

2.4 Task executor (EMCTASK)

EMCTASK is a task level command handler and program interpreter for the RS-274 NGC machine tool programming language [24], commonly referred to as G code. It monitors the status of subordinate modules (EMCMOT and EMCIO) and coordinates them. It also receives and analyzes commands, either from the operator through GUI or from another process (locally or remotely in both cases), interprets them as NML messages and dispatches to EMCMOT, EMCIO or EMCTASK itself at appropriate times.

The actual commands, written in the form of G and M code programs, can be sent to EMCTASK using the Machine Device Interface (MDI) mode or as a file when the machine is in Auto mode. As interpreter for G and M code programs, whose coding does not vary significantly between machines, it is less machine specific than the EMCIO.

2.5 User interfaces

Several user interfaces have been developed for EMC2: *keystick*, *xemc*, *tkemc*, *mini* and *AXIS*. All of these programs natively run under a Linux-based OS and all run in the X11 environment (X Window System), with the exception of *keystick*, which is text-based. *AXIS* is the most advanced GUI, featuring an interactive G-code previewer. GUI-based programs can be expanded and adapted to match specific application needs by means of virtual control panels (VCP), which is supported with pyVCP package. Besides the above-mentioned user interfaces, a telnet-based program emcrsh is also provided for running remote sessions. Multiple GUIs can be run simultaneously, either locally or remotely, across multiple networked computers. Several possibilities are presented for remote connection: utilization of X11 protocol, VNC connection, or running GUIs natively on a remote computer.

In the first case, GUI runs on the main PC, and the keyboard, mouse and GUI window are forwarded to the remote PC. Installation of X server software on remote computer is required for X11 connection such as Xming for Microsoft Windows™ operating systems. VNC connection is remote desktop system, transferring control of main PC to remote PC. X11 and VNC connections can be tunneled over Secure Sockets Layer (SSL), in order to provide network security during sessions (X11 over SSL, VNC over SSL). In the third case, GUI programs run natively on remote PCs and communicate with main PC by exchanging NML messages over the network. This configuration requires modification of configuration files on the main PC (adding appropriate remote IP addresses fields in NML configuration files).

All of above-mentioned GUIs will work in this configuration under Linux-based operating systems. Tkemc and mini can be run under Mac OS-X™ or Microsoft Windows™ platforms, if Tcl/Tk programming language, in which these GUIs are coded, has been installed.

Several GUIs are also provided as tools for configuration and tuning during machine setup in real-time. *ClassicLadder* is used for graphical editing of ladder logic diagrams, *HALConfig* is a tool for testing and parameterization of machine configuration, *HALScope* is a software oscilloscope for monitoring HAL signals and *HALMeter* is a simple tool for monitoring individual HAL signals. Additional GUI packages are also being developed, and will provide support for graphical building of HAL configurations and machine motion visualization.

2.6 Compatible hardware interfaces

As previously discussed, hardware interfaces which connect the EMC2 with feed drives must execute in real time, while the remaining interfaces can execute in the non-real-time environment. Many existing interface cards available on the market lack the necessary drivers and/or documentation, which limit their integration into the EMC2.

Available hardware interfaces for the EMC2 are mostly ISA / PCI / PCI Express bus-based cards, with a few low-end solutions available for the LPT port. This currently limits the installation of the EMC2 to desktop computers with such buses. In this project, only PCI/PCI Express interfaces, for which suitable motherboards can be obtained on the market, are considered.

Presently, support for absolute encoder (EnDat) or FieldBus interfaces (ProfiBUS, CANOpen, DeviceNET) is still not implemented in the EMC2, which further limits its application possibilities to modern feed and main spindle drives. Recent user contributions, however, provide some limited support for MODBUS and EtherCAT, which is also not yet integrated into the EMC2 source tree. RTAI Linux drivers for the CAN PCI interface card produced by Peak System GmbH, as well as non-real-time community drivers for acquisition cards produced by National Instruments and Measurement Computing, could easily be integrated into the EMC2.

Available interface support is currently limited to general purpose analogue/digital I/O circuitry, which can be used for a simple interfacing stepper, variable frequency drives or servo systems with incremental encoders. These interfaces can be used to transmit speed/position reference signals to the feed or the main spindle drives in one of common forms (Step + Direction, PWM + direction, and ± 10 V) and to receive encoder feedback signals. Such interface PCI cards, which have been used in this project, include a Motenc-lite card manufactured by Vital system Co. and Anything IO cards manufactured by Mesa Electronics Co.

With all breakout boards installed, the single Motenc-lite card features 8 analogue outputs (Range ±10 V, 13-Bit Resolution), 8 analogue inputs (Range ±5 V, 14-bit Resolution), 4 Differential Quadrature Encoder Inputs (DQEI), as well as 16 digital outputs (24VDC) and 32 digital inputs (24VDC). The DQEI interface is mapped to internal 32-bit up/down counter with maximum update frequency of 2 MHz. Multiple installations (up to four) of this card are possible in order to further expand the configuration.

All Anything IO card models are based on the Field Programmable Gate Array (FPGA) hardware, which is end-user accessible through the available open source driver software, allowing versatile configurations and interfaces. Depending on the available model and corresponding daughter boards, anything I/O card configurations may vary from the generic digital I/O to the multi-channel analogue servo interface, consisting of DQEI and DAC output per channel. Configuration of up to 16 analogue servo interfaces is possible with a single anything I/O card model 5i22. Compatibility with the EMC2 software is provided through the HostMot2 firmware [19]. The EMC2 support for the HostMot2 is split into a generic driver for configuring the card and low-level I/O drivers for accessing anything I/O boards from the bus side.

3 Experimental machine tool

A large outdated planning machine, manufactured by "KovositHoloubkov" company in 1959, was used in the development of a milling testbed platform (Fig. 3). The selected machine was considered to be a suitable base because its support elements and guides provide sufficient rigidity for the medium and light machining conditions. However, extensive redesign of the most of remaining parts was mandatory in order to achieve an open CNC machine tool (Fig. 4).

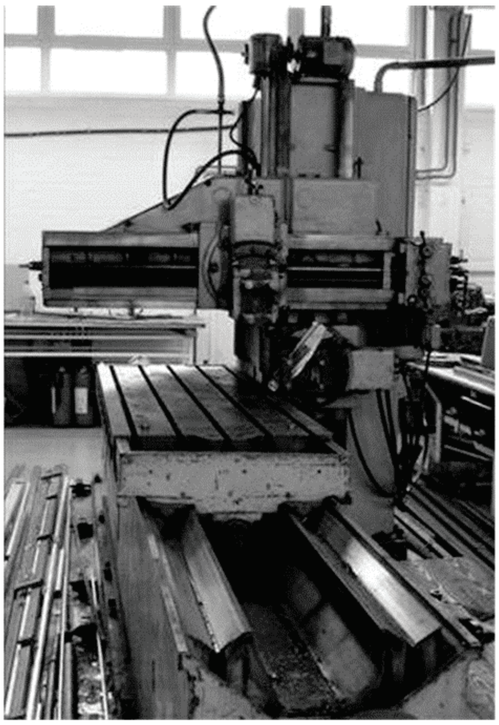
Figure 3 Planning machine used as a base for experimental machine tool development

The redesign approach was aimed at providing a basic 3-axis Cartesian kinematic structure, with an upgrade possibility to a 5-axis machine.

3.1 Control cabinet

The design of control cabinet was focused on a solution that would be as generic as possible (Fig. 5). This approach was used in order to provide a control system suitable for a future prototype development of machine tools with various configurations.

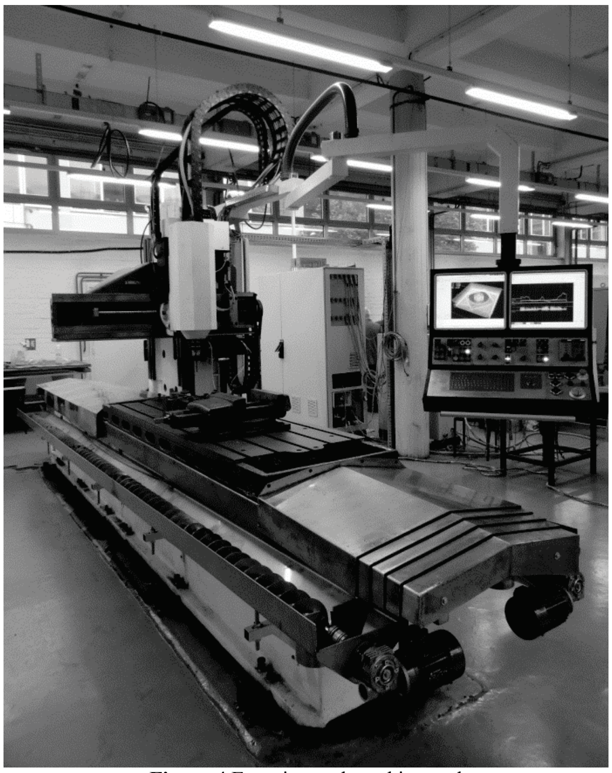
Figure 4 Experimental machine tool

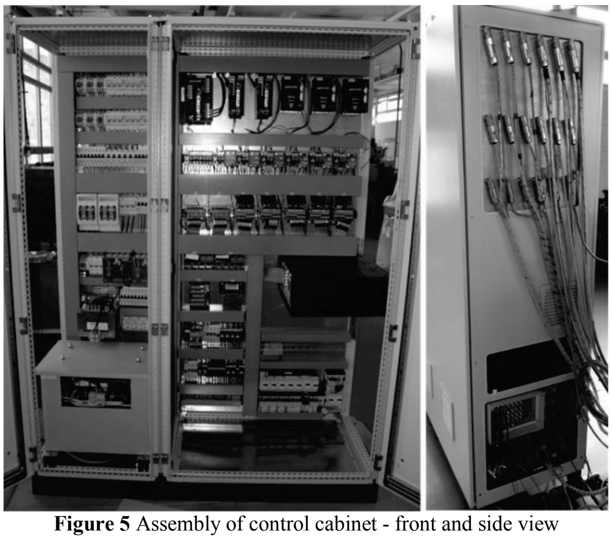
Figure 5 Assembly of control cabinet - front and side view

Presently, it is capable of controlling up to six servo axes and two main spindles. A matching set of connectors for motor power, encoder and limit switches is installed for each feed drive allowing their fast reconfiguration. In addition, a set of I/O connectors is also installed to provide:

- Torque output signals from feed drives and main spindle,
- General purpose analogue and digital I/O,
- Firewire/USB/Ethernet/RS-232 connections.

3.2 Feed Drives and Main Spindle Unit

Basic technical information on the machine feed drive configuration is presented in Tab. 1.

Design and assembly of *X*-axis involved replacement of a hydraulic cylinder with a 2500 mm long ball screw assembly attached to machine table (2200 × 800 mm).

Suitable bearing support elements also had to be made and facing of matching areas in the machine bed had to be performed so that they could be mounted. Existing guides were originally designed for translational main motion velocities up to 25 m/min. They were in a good condition with no visible signs of wear.

Table 1 Technical characteristics of feed drives

Parameters	Axis			
	X	*Y*	*Z*	*Z′*
Travel (mm)	2200	750	200	600
Rapid traverse (m/min)	20	7	7	-
Max. feedrate (m/min)	12	7	7	-
Resolution (μm)	0,5	0,5	0,5	-

Regarding the *Y*- and *Z*-axis, existing drives were both replaced with suitable ball screw assemblies and corresponding belt transmissions. Dovetail guides were left unchanged, in order to preserve axis rigidity, even though minor wear was locally present and guide design allowed relatively low velocities of up to 7 m/min. Belt transmission was necessary for both axes in order to fully utilize available feed motors. Original machine design also featured auxiliary vertical support drive (*Z′*-axis) used for rough tool height adjustment in case of higher workpieces. At present, this drive, consisting of AC induction motor with worm gear and trapezoidal screw assembly, is left intact.

The control of feed drives is based on a closed-loop servo system, using permanent magnet synchronous motors (PMSM) with integrated incremental encoders and corresponding motor controllers. Digital servo controllers, type DPCANIE-030A400, are used for driving the *Y*- and *Z*-axis motors, while DPCANIE-060A400 is used for driving the *X*-axis motor. Three additional controllers, type DPCANIE-015A400, were installed in order to provide a possibility for the feed drive upgrade.

The selected drives provide multiple modes of operation (closed-loop control of position, velocity or torque, as well as encoder following for electronic gearing), a variety of common industrial interfaces for the acquisition of reference signals (±10 V, PWM+Direction, STEP+Direction), a CAN-bus interface with a CANOpen protocol for fieldbus connections, as well as multiple general purpose analogue and digital I/O ports. Diverse interface possibilities make these drives a practical choice for both industrial and research purposes.

Interface to the feed and main spindle drives from the PC side is done via a suitable set of both the Motenc-lite and the 5i20 and 5i22 anything I/O PCI cards. Any IO PCI cards are mostly used for interfacing servo drives with ±10 V speed reference analogue signal and DQEI as velocity/position feedback. Remaining I/O resources of all cards are used for the real-time data acquisition and the general purpose I/O.

In the current setup, feed drives are configured as closed-loop velocity controllers because the position loop for each axis is implemented within HAL by instantiating the real-time PID modules. Controller gain factors for both the current and velocity loops are obtained manually, using a royalty-free DriveWare configuration software package. Gain factors for position loops are also obtained by manual tuning using the EMC2 tuning utilities. The main spindle unit, type HSM105SE-V01-FK719-11 (28 kW, 63 N·m, 12 000 rpm), and its control system were designed and produced separately by HSTec Co. The control system is also connected through HAL pins with the EMC2.

Fig. 6 illustrates detailed configuration of HAL for presented machine tool, as well as essential wiring schematics for feed drives and main spindle. Each controlled axis with corresponding drive and HAL signals is denoted by *i*, where *i* = [*X*, *Y*, *Z*]. Allocated hardware I/O ports are denoted as DI (digital inputs), DO (digital outputs), ADC (A/D converters) and DQEI.

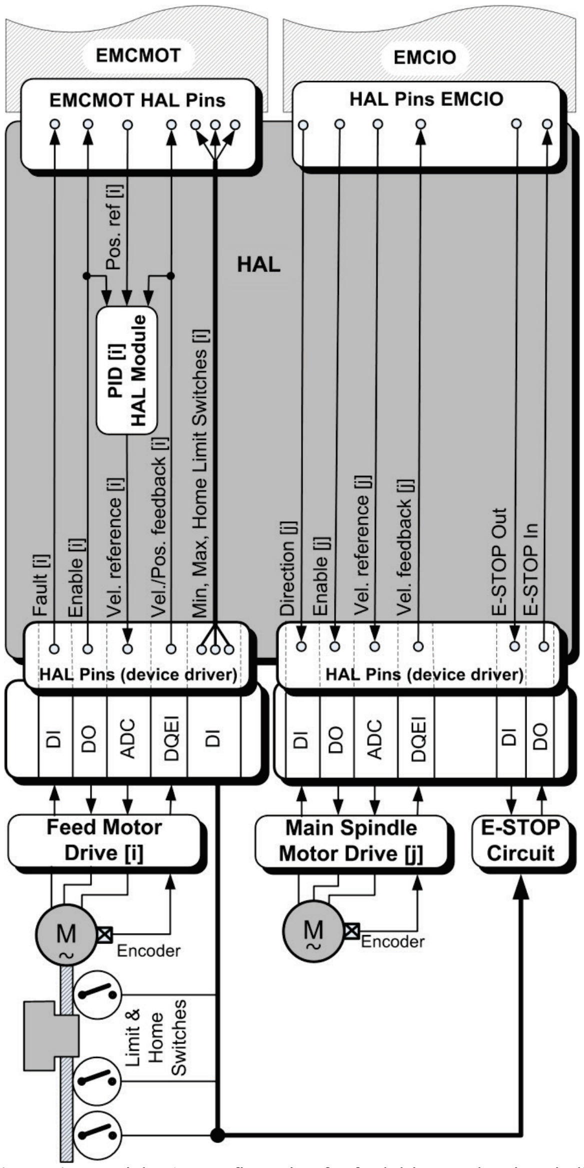

Figure 6 Essential HAL configuration for feed drives and main spindle

The main spindle unit is partly reconfigured so that it can be upgraded with the spindle monitoring system (Dittel Messtechnik Gmbh). The measuring system has a sensor for the shaft displacement detection, temperature and vibration sensors in the front bearing, a temperature sensor in stator windings, a crash sensor, a tool clamping sensor and a data logger, which connects monitoring system with the control unit or external PC using RS485 connection.

3.3 Auxiliary systems and operator panel

Existing lubrication system, originally used for the main drive, was not adequate for numerically controlled *X*-axis guides because continuous (back and forth) full guide travel motion was required for its proper operation. Remaining guides were cord lubricated, supplied by small oil pools placed in corresponding locations. Circulating oil with constant flow of 2 l/min, invariant of axis travel, was required for lubrication of *X*-axis guides.

Impulse lubrication, cycle-dependent on travel distance in given period of time, was required for other axis. Centralized lubrication system was designed to specifically match described lubrication requirements for retrofitted feed drives. For this purpose, customized non-real-time HAL module was developed, which monitors each axis travel distance over HAL signals connected to DQEI channels and triggers lubrication cycles accordingly.

Proper operation of the main spindle unit required the design of two additional subsystems – spindle cooling system and tool/workpiece clamping aggregate. Cooling system was designed to automatically maintain constant temperature of the main spindle unit with respect to rated power output. Hydraulic clamping aggregate, in addition to tool clamping, supports up to three cylinders for workpiece clamping. Customized non-real-time HAL module was also developed to control tool change and clamping system using an existing generic EMC2 tool change interface.

Design of NC operator panel aimed to provide versatile practical solution for industrial, research and educational purposes (Fig. 7).

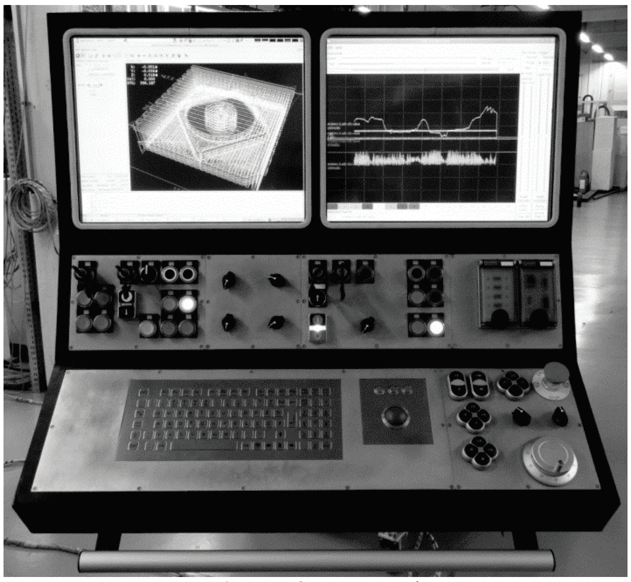
Figure 7 Operator panel

Dual touch screens provide convenient human-machine interface (HMI), allowing visualization of process variables in real time and user interaction through virtual control panels (VCP). In addition, industrial (IP67) PC keyboard with trackball mouse was installed for easier data input. Standard industrial pushbuttons were installed for control of the main spindle, clamping system, flood/mist, chip conveyer, vertical support drive, etc. Overrides for angular and linear feed rates, rapid traverse and spindle speed are available through BCD coded rotary switches. Control of up to six axes by jogging or by manual pulse generators (MPG) can be done directly from NC panel or through attached remote control unit. All pushbutton/rotary switch controls are connected to the PC running EMC2 by set of four 7i64 isolated I/O cards, produced by Mesa Electronics Ltd, over the USB port. Customized non-real-time HAL module was specifically developed to provide support for those cards by providing necessary HAL pins.

4 Conclusions

Implementation of the EMC2 (V2.3), a Linux-based OAC system, in a three-axis industrial milling machine, together with the characteristics of its redesigned structure and operational capabilities has been presented in this paper. Although the current machine tool setup used in this study cannot be used to present the full potential of the EMC2 system, some key features are emphasized.

The EMC2 software fulfils important research and educational requirements due to the benefits of robust open source real-time OS, including expansion possibilities with other open-source programs. The lack of device driver support from most hardware vendors, especially in the area of fieldbus interfaces, still remains the main limitation of this system. This support would allow advanced interfacing of EMC2 with various up-to-date drive and PLC systems. However, the development of EtherCAT support, which relies on a common on-board network interface, shows promise and could easily remove this limitation. Besides the aforementioned, there are already compatible hardware interfaces, developed by several manufacturers, for both servo and stepper drive systems.

The EMC2 configuration can be easily adapted to match any common machine structure with a standard set of auxiliary functions (e.g. lubrication, mist/flood, etc.). However, apart from a proper understanding of the EMC2 system itself, the implementation of such system may also require multidisciplinary knowledge in the field of machine tool design, hardware interfaces, CNC control, Linux OS, versatile programming skills which include certain OS modifications, C/C++, Python and other skills required for realization of the desired machine tool characteristics. For more complex multi-axis machinery with advanced auxiliary systems, the extensive coding of machine specific modules in the EMC2 is mandatory.

5 References

[1] Pritschow, G.; Daniel, Ch.; Junghans, G.; Sperling W. Open System Controllers - A Challenge for the Future of the Machine Tool Industry. // CIRP Annals - Manufacturing Technology. 42, 1(1993), pp. 449-452.
[2] Asato, O. L.; Kato, E. R. R.; Inamasu, R. Y.; Porto, A. J. V. Analysis of Open CNC Architecture for Machine Tools. // Journal of the Brazilian Society of Mechanical Sciences. 24, 3(2002), pp. 208-212.
[3] Koren, Y.; Jovane, F.; Heisel, U.; Moriwaki, T.; Pritschow, G.; Ulsoy, A. G.; van Brussel, H. Reconfigurable Manufacturing Systems. // CIRP Annals – Manufacturing Technology. 48, 2(1999), pp. 527-540.

[4] Proctor, F. M.; Michaloski, J. L. Enhanced Machine Controller Architecture Overview, 1993, from ftp://ftp.isd.mel.nist.gov/pub/NISTIR_5331.pdf, accessed on 2012-12-16.
[5] Pritschow, G.; Altintas, Y.; Jovane, F.; Koren, Y.; Mitsuishi, M.; Takata, S.; van Brussel, H.; Weck, M.; Yamazaki, K. Open Controller Architecture – Past, Present and Future. // CIRP Annals – Manufacturing Technology. 50, 2(2001), pp. 463-470.
[6] Erol, N. A.; Altintas, Y.; Ito, M. R. Open system architecture modular tool kit for motion and machining process control. // IEEE/ASME Transactions on Mechatronics. 5, 3(2000), pp. 281-291, DOI: 10.1109/3516.868920.
[7] Hecker, R. L.; Liang, S. Y. Power Feedback Control in Cylindrical Grinding Process. // Proc. ASME International Mechanical Engineering Congress and Exhibition, 2000, pp. 713-718.
[8] Schofield, S.; Wright, P. Open Architecture Controllers for Machine Tools, Part 1: Design Principles. // ASME Journal of Manufacturing Science and Engineering. 120, 2(1998), pp. 417-424.
[9] Wright, P. K.; Dornfeld, D. A. Agent-Based Manufacturing Systems. // Trans. of North American Manufacturing Research Institute – SME. 24, (1996), pp. 241-246.
[10] Park, J.; Pasek, Z. J.; Birla, S.; Yansong, S.; Koren, Y.; Shin, K. G.; Ulsoy, A. G. An Open Architecture Testbed for Real-Time Monitoring and Control of Machining Processes. // Proc. of American Control Conference, 1995, pp. 200-204.
[11] Rober, S. J.; Shin, Y. C. Modeling and Control of CNC Machining Using a PC-Based Open Architecture Controller. // Mechatronics. 5, 4(1995), pp. 401-420.
[12] Altintas, Y.; Munasingh, W. K. Modular CNC Design for Intelligent Machining, Part 2: Modular Integration of Sensor Based Milling Process Monitoring and Control Tasks. // ASME Journal of Manufacturing Science and Engineering. 118, 4(1996), pp. 514-521.
[13] Koren, Y.; Pasek, Z. J.; Ulsoy, A. G.; Benchetri, U. Real-Time Control Architectures for System Performance. // Annals of the CIRP. 45, 1(1996), pp. 377-380.
[14] Suh, S-H.; Kang, S-K.; Chung, D-H.; Stroud, I. Theory and Design of CNC Systems, Springer-Verlag, London, UK, 2008.
[15] Liang, S.; Hecker, R. L.; Landers, R. G. Machining Process Monitoring and Control: The State–of–the–Art. // ASME Journal of Manufacturing Science and Engineering. 126, 2(2004), pp. 297-310.
[16] EMC applications, from wiki.linuxcnc.org, accessed on 2012-12-16.
[17] EMC User Manual, from http://www. linuxcnc.org, accessed on 2012-12-16.
[18] RTAI - the Real Time Application Interface for Linux, from http://www.rtai.org, accessed on 2012-12-16.
[19] RT Linux, from www.rtlinuxfree.com, accessed on 2012-12-16.
[20] Proctor, F. M.; Shackleford, W. P.; Michaloski J. L. The Neutral Message Language: A Model and Method for Message Passing in Heterogeneous Environments, 2000, from http://www.isd.mel. nist.gov, accessed on 2012-12-16.
[21] EMC Components, from http://wiki. linuxcnc.org, accessed on 2012-12-16.
[22] HAL User Manual, from http://www.linuxcnc.org, accessed on 2012-12-16.
[23] EMC Integrator Manual, from http://wiki.linuxcnc.org.
[24] Marietta, M. Next generation controller (NGC) specifications for an open system architecture standard (SOSAS). Technical report - National Center for Manufacturing Sciences, Denver, USA, 1994, from ftp://ftp.isd.mel.nist.gov/pub/NGC_document.pdf, accessed on 2012-12-16.

Authors' addresses

Tomislav Staroveški, assistant
Department of Technology
Faculty of Mechanical Engineering and Naval Architecture,
University of Zagreb
Ivana Lučića 5, 10000 Zagreb, Croatia
E-mail: tomislav.staroveski@fsb.hr

Danko Brezak, assistant professor
Department of Robotics and Production System Automation
Faculty of Mechanical Engineering and Naval Architecture,
University of Zagreb
Ivana Lučića 5, 10000 Zagreb, Croatia
E-mail: danko.brezak@fsb.hr

Toma Udiljak, professor
Department of Technology
Faculty of Mechanical Engineering and Naval Architecture,
University of Zagreb
Ivana Lučića 5, 10000 Zagreb, Croatia
E-mail: toma.udiljak@fsb.hr