Control System For The Waterjet Cutting Machine

A. Hace, K. Jezernik, Member, IEEE

Abstract—This paper addresses the field of factory automation within the scope of machine control. It presents the control system for a waterjet cutting machine. It involves a PC-based CNC controller that has been built on a QNX real-time operating system platform. Microsoft Windows, which is a commonly accepted graphical user interface environment, serves as a HMI front-end. The CNC controller and the HMI are interconnected by the Ethernet TCP/IP link. The control system is flexible and modular. It supports CAD/CAM and a network link to the office-level of factory automation. The application is thoroughly presented with the highlight on real-time features; the QNX real-time operating system is analyzed in detail. Moreover, Ethernet TCP/IP technology for the use in a networked automated factory is also put in the foreground.

Index Terms—Factory automation, CNC, Ethernet, Real-time operating systems, Control applications.

I. INTRODUCTION

In recent years, PC-based control technology has become a widely used industry practice. Benefits include faster design cycles, lower downtime using diagnostics and simulation tools, increased productivity and decreased maintenance costs. Moreover, open system designs that use standard hardware and operating system software minimize cost, permit system scalability, and ensure future performance enhancement. However, hardware-based CNC systems still dominate the world of machining control.

This paper deals with a PC-based CNC system. In order to implement CNC functionality on a PC-platform, it shall operate in real-time. A good real-time operating system is therefore an unavoidable choice. Well-known operating systems like Windows or Linux can be used for PC-based control of mechatronic systems, if these operating systems are modified with real-time extensions [1], [2], [3], [23], [26]. Special attention among academic researchers is given to Real-Time Linux [12], which implements a dual-kernel model with real-time tasks (RT tasks) running in kernel-space and supervisor mode [13]. Although there is clear evidence of successful implementations using Real-Time Linux, with the creation of a commercial version and the filling of a controversial patent the support of the community dropped. Furthermore, the use of Real-Time Linux is not always justified in technical terms [19]: lack of determinism for existing Linux applications and drivers, duplicated coding efforts since RT tasks can not make full use of existing Linux system services, fragile execution environment since RT tasks do not benefit from the robust Linux MMU-protected memory space, limited functionality and portability since the RT kernel provides only a subset of the services provided by standard POSIX and Linux APIs. These are main reasons why a more advanced and feature-rich real-time operating system should be used.

The QNX real-time operating system (RTOS) can be a good candidate for use in the field of robotic and automation applications [5], [6], [8], [7], [14]. The RTOS microkernel is surrounded by a collection of optional processes that provide POSIX services. Its open and scalable architecture is robust and reliable, allowing the implementation of hard real-time control algorithms as well as networking and graphical user interface on a standard low-cost PC platform. QNX also features a rich set of development tools and good technical support that make applications easy to develop and maintain.

Open architecture systems need an effective network in order to be capable of supporting interoperability of control devices in a distributed automation environment. Profibus, DeviceNet, ControlNet, and CAN, are well-known open standard industrial interfaces being used in the field of factory automation. In an office environment the well-known Ethernet has been largely utilized since it provides a fast data path of extremely high bandwidth at a very low cost. Thus, Ethernet has also been moved to production levels of automated factories. Recently, however, Ethernet networks have gained the capability of deterministic communication in real-time [15], [24], [25] that allows their use at all levels in a factory. Thus, Ethernet is a common data network base technology whereto most parts of the industrial automation industry are rapidly converging [9], [18], [22].

The aim of this paper is to present case application of the CNC system that has been developed for a waterjet cutting machine. The application involves a CNC controller and a graphical user interface that are implemented on a networked PC-based system. Ethernet TCP/IP realizes the network link; the GUI uses Microsoft Windows as the front-end. QNX has been used as a platform for implementation of the CNC control...
algorithms in order to achieve hard real-time performance. Section 2 focuses on the QNX real-time operating system and also reveals some important details of a modern RTOS architecture. Section 3 presents arguments about the use of the Ethernet for factory automation. Section 4 describes the case application, section 5 shows main results, and section 6 concludes the paper.

II. QNX REAL-TIME OPERATING SYSTEM

QNX is a multi-process 32-bit operating system with an advanced system architecture [21]. It is a tiny RTOS consisting of a lean microkernel, which is mainly responsible for process scheduling, interprocess communication (IPC), low-level network communication, and first-level interrupt handling, thus having a small API with only few system calls. All other operating-system functions such as device I/O, network management etc. are handled by modules, which are dynamically loaded when needed. System processes and device drivers are essentially not different from user processes. Moreover, the QNX architecture provides a memory-protected address space for all software components (Fig. 1), including operating system modules, drivers and application processes, which makes QNX safe and reliable. As a price, real-time performance is not seriously deteriorated. Its IPC network capabilities are of a great advantage: processes communicate with each other via messages, whether they reside locally or on remote nodes. Ethernet, which has become an industry standard, is also supported. Moreover, QNX add-ons offer full implementation of the Internet Protocol suite, including various popular network TCP/IP applications and libraries for writing programs.

A. Client/Server Architecture

Mechatronic control systems are commonly based on a processor, which utilizes interface hardware (I/O board). Software device drivers operate these boards. Traditional device drivers generally reside in an operating system's kernel (Fig. 2a). Hence, difficulties to write and maintain are encountered in Microsoft Windows NT [3], [26] as well as in RT Linux [2], [13]. In addition, accessing a kernel-mode device driver from a user-mode program requires a system call, which causes an overhead. To overcome this problem, device drivers may be linked to the user's control program. This method is simpler and more efficient, since there is no need for a system call in the kernel. However, it is also less secure. The device interface library must access hardware directly; therefore, the...
user's control program must have privileged access and, hence, is capable of crashing or corrupting the entire system. Additionally, multiple programs may interfere while simultaneously attempting to communicate with the hardware board; consequently, only one control program may access the hardware at any time. The QNX open architecture allows for avoiding these problems. Programs that serve the purpose of device drivers run in user mode. A user control program uses IPC to communicate with the device driver. Device drivers that serve the requests from multiple user programs are called hardware servers. The user programs that use the server to communicate with the hardware are called clients. The client/server architecture is illustrated in Fig. 2b. This architecture allows for easier development of the hardware server that is less complex than a device driver. Since servers can be stopped and started at any time they are also easier to configure. Besides generic clients can be independent of the specific hardware board. Furthermore, multiple clients can connect to the same hardware without interfering with each other. The client program can be even located on a different machine from the server program. However, in this case, the network should be deterministic and fast enough for the transfer of I/O data.

B. Real-time multiprocess environment

A hard real-time computation system must run an operating system, which given sufficient computational capacity guarantees that a feasible schedule can be executed. That means if the environment of the system is under control, the operating system itself will not be the cause of non-timeliness computations. In order to do so, the following basic requirements must be met:

- Always execute higher-priority tasks in preference to lower priority tasks.
- Priority inversions, which may result when a higher-priority task needs a resource allocated to a lower-priority one, are bounded in time.
- Operating-system activities, which can not be scheduled, do not exceed the remaining capacity in any particular division.

A real-time application must respond deterministically and the response must be predictable. Hence, a proper schedule algorithm should be applied. In order to comply with the requirements above, a real-time operating system/kernel shall support fixed-priority preemptive scheduling for tasks, and provide priority inheritance or priority-ceiling emulation for synchronization primitives. The kernel itself must be pre-emptable as well. Moreover, it must guarantee interrupts having a fixed upper bound on latency; by extension, nested interrupt support is required. Executing operating-system services at a priority determined by the client of the service must be also supported. Furthermore, priority inversion avoidance must be applied to all shared resources used by the service.

The QNX complies with all the demands above. Every process is assigned a priority. The process with the highest priority is dispatched to CPU. QNX provides FIFO scheduling and round-robin scheduling. It also implements a pre-emptive kernel. Most transactions between processes follow a client/server model. Servers provide some form of service and clients send messages to these servers to request a given service. Usually the number of clients is higher than the number of servers. As a result, a server will likely run at a priority that exceeds the priorities of all its clients and that can cause priority inversion. Therefore QNX implements priority inheritance mechanism: the server inherits the priority of its client process. It is also important to minimize the time from the occurrence of an external event to the actual execution of a code. The interrupt handler can simply return, or return and cause a kind of short message to be triggered that keeps the delay as small as possible. Higher-priority interrupts can pre-empt a lower-priority interrupt. QNX fully supports the PC interrupt mechanism: interrupt handlers as well as attached interrupt processes are accordingly pre-empted.

III. ETHERNET TCP/IP IN FACTORY AUTOMATION

The case application in this paper deals with the device level and cell level in a factory automation network (Fig. 3). At the device level short record data is exchanged between controllers, which are usually microprocessor boards and/or PLCs, sensors and actuators in real-time. The data is exchanged cyclically or asynchronously. In the first case, the device controller reads variables from the sensor and sends commands to the actuators. In second case, an alarm generated by a device interrupts the cyclic data exchange for the controller to be alerted. At the cell level the coordination of the device controllers involves the download and upload of data for configuration, calibration and monitoring purposes. The record of exchange data is rather long and commonly has a structured form, termed object. The objects are transferred according to the client-server model and the allowable transmission times can be several hundreds of milliseconds. Typical application is Human Machine Interface - HMI.

Ethernet can be applied for factory automation network. It has been specified by the IEEE 802 Committee as a full-duplex operation network connection - a single node is either transmitting or receiving at any instant. Internet protocols TCP (Transmission Control Protocol), UDP (User Datagram Protocol), and IP (Internet Protocol) are commonly linked with
Ethernet (Fig. 4). For interoperability among different network devices, a common application layer (layer 7) is required. Application layer protocol defines the nature of a message. Although the CSMA/CD (Carrier Sense Multiple Access / Collision Detection) scheme makes the Ethernet protocol non-deterministic and unsuitable for real-time applications, recent performance improvement with a higher transmission speed (up to 10 Gbit/s in near future) and switches make Ethernet proper for use in real-time applications. The switch is able to recognize the addresses of the stations connected to it and to redirect the messages only to the destinations. To avoid collisions the buffered switch stores the concurrent messages and delivers them in sequence to the destination soon after. Switches may have support for priority, where the high priority and delivers them in sequence to the destination soon after. Collisions the buffered switch stores the concurrent messages and to redirect the messages only to the destinations. To avoid collisions the buffered switch stores the concurrent messages and delivers them in sequence to the destination soon after. Switches may have support for priority, where the high priority queues are reserved for real-time critical data. Such full-duplex priority-based switched Ethernet can be deterministic and can support the real-time network.

![Fig. 4: The 7-layer OSI Network Model and UDP/TCP/IP stack](Image)

**Fig. 4: The 7-layer OSI Network Model and UDP/TCP/IP stack**

<table>
<thead>
<tr>
<th>Application</th>
<th>Protocols/services for applications</th>
<th>FTP, FTP, Telnet, HTTP, etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Presentation</td>
<td>Selecting of type of dialog</td>
<td>Identification and Authorization</td>
</tr>
<tr>
<td>Session</td>
<td>Representation of data</td>
<td>Control of start/end of transmission</td>
</tr>
<tr>
<td></td>
<td>Definition of coding type</td>
<td>Error detection and clearing</td>
</tr>
<tr>
<td>Transport</td>
<td>Sequencing of application data</td>
<td>UDP/TCP</td>
</tr>
<tr>
<td></td>
<td>Control of start/end of transmission</td>
<td>Setup/release of connections</td>
</tr>
<tr>
<td>Network</td>
<td>Routing, prioritization</td>
<td>Flow control</td>
</tr>
<tr>
<td></td>
<td>Framing</td>
<td>IP</td>
</tr>
<tr>
<td>Data Link</td>
<td>Framing</td>
<td>Ethernet</td>
</tr>
<tr>
<td>Physical</td>
<td>Bit transmission</td>
<td>TCP</td>
</tr>
<tr>
<td></td>
<td>Coding</td>
<td>Real-time protocol</td>
</tr>
<tr>
<td></td>
<td>Synchronization</td>
<td>UDP</td>
</tr>
</tbody>
</table>

**Table 1:** OSI layer protocols/services

**Fig. 5:** Ethernet communication profile in the field of factory automation

The waterjet cutting machine has been built for cutting of leather or synthetic textile in shoe industry. The main parts of the machine are the transport system, the XY system with the cutting head, and the high-pressure (HP) pump (Fig. 6). The transport system feeds material into the cutting section and holds it during the cutting period. It consists of three sections. In the input section the material is laid down on a transport table. The table then transports material toward the cutting section. When cutting, the gripper holds down the material and the transport system is stopped. Afterward, the cut pieces are transported to the output section. Electrical motors and pneumatic cylinders power the transport system. The XY system is a two-axis table with both axes perpendicular to each other. It drives the cutting head in a horizontal plane. The axes are implemented by ball screw drives with high performance electrical servomotors. The HP pump supplies the cutting head with water at high pressure. The cutting nozzle on the cutting head is linked to the HP pump by a high flexible tube.

The machine can operate in a manual or automatic mode. However, the fundamental operation of the machine is within the automatic cycle. The automatic cycle encompasses two main operation phases: transport of material and cutting. Cutting is accomplished on the base of a NC program generated by CAD/CAM software. The program can be changed in each cycle.

**B. The embedded control system**

The embedded control system performs various diverse tasks needed for a CNC system: time-critical CNC servo control of position loops, acceleration/deceleration, NC path generation, program loading and interpreting, graphical user interface, file management, data processing, and networking. The
single-processor QNX platform can run hard real-time control algorithms as well as a bulk of soft and non real-time software modules due to the deterministic response of the operating system. This architecture replaces the traditional multiprocessor host/DSP board architecture often used in control applications, which suffers from the high cost of the hardware, the limited flexibility of the system, and the complexity of the software. Advantages of a single-processor system include reduced costs and lower complexity, as well as increased flexibility and upgradability. The use of modern high-speed consumer-grade PCs coupled with a real-time operating system allows the development of a system that implements the control algorithms (programming, trajectory planning and serving), networking, and user interface on one CPU. However, in order to achieve interoperability of the already existing CAD/CAM software, our application was designed by the two-processor architecture:
- The Windows computer implements the GUI.
- The QNX computer serves for CNC functionality.

The CNC package comprises the XY control algorithms, the basic water jet control routines, the operator level control procedures, and the communication modules. The end-user installation of the CNC controller excludes a keyboard/mouse and a display device connection. The Windows computer implements the GUI as well as hosting CAD/CAM software with a connection by an additional Ethernet link to the factory LAN. The GUI connects user input/output devices. Standard Fast Ethernet interface cards interconnect both PCs. The control interface, which utilizes the high-performance multifunction I/O board for PC motion control applications [20], connects the CNC controller with the XY system and the PLC that controls the transport system and the HP pump. The control board is linked with XY servo drivers as well as with the external PLC. The PLC main tasks are to control the transport system and the HP pump. The RS232 serial line connects the CNC controller and the PLC.

C. The graphical user interface

The GUI brings together all of the displays and functions for machine management. It has been developed in order to provide user-friendly machine operation control to allow for easy diagnostics and maintenance. All the relevant machine information as:
- general system information that includes machine operation mode (auto/manual), execution mode (running/stopped) and other general machine state information (machine homed),
- feedrate information that includes commanded, actual and override information,
- axis information that includes position and other axis status information,
- program information (program block, program name, program status),
- tool information (jet start/stop),
- PLC information (automatic operation stopped, started, running),
- auxiliary devices information (HP pump state, mains state, safety circuit state),
- miscellaneous information (emergency stop pressed, network alive, reset fault),
- message information that includes various levels of alarm and operator message information,

are displayed on the monitor desktop. The GUI provides monitoring, programming, configuring, and multifunction software-button support (Fig. 8). Furthermore, it contains the contour geometry field that occupies the central area of the screen. It shows up a whole program with a planned cutting contour when loaded. However, during program execution the cutting head path is redrawn in real-time while traveling. Contour segments that are cutting and those that have been already cut are marked differently. The actual position of the cutting head on the planned cutting contour is marked with a cross.

![Fig. 7: The control system block diagram](image)

![Fig. 8: The Graphical User Interface](image)
machine-control procedure is started. It displays all the procedure steps, which are to be executed to accomplish the procedure. The operator is guided through the procedure; instructions are displayed for manual interventions if necessary. This approach makes the machine operation transparent. Consequently, the potential fault can be clearly identified by messages displayed on the GUI and the operation can be recovered quickly. Error messages are displayed in an easily understandable and correct manner. Furthermore, the operator is able to display a message history list to address an alarm condition. Meanwhile, the GUI continues to monitor and record all new alarm events. The machine state information is acquired cyclically at relatively slow refresh periods with respect to the fast position measurement; the cycle rate however, shall be less than a second in order to ensure prompt machine response and to serve messages as they occur.

D. The CNC controller

Fig. 9: The CNC software structure

The CNC performs various tasks necessary for machine-control: basic control logic procedures, time-critical position control loops, acceleration/deceleration, NC path generation, program loading and interpreting, machine programming interface, graphical user interface, file management, data processing, and networking. Additionally, it provides for interoperability with the external PLC in order to synchronize operation of the XY system, the transport system, and the HP pump. All user-important information related to the machine must be passed to the operator. Thus, the CNC controller supplies the GUI with the position/speed of the cutting head, G-code program execution information, messages and alarm conditions. Furthermore, the controller forms an information link that connects the external PLC with the GUI. The link enables the transfer of the transport system and the HP pump operation data. In order to accomplish all the tasks cited above, the CNC software package architecture must allow for real-time operation: hard real-time requirements are associated with motion control algorithms while the GUI messaging scheme requires soft real-time execution. The client/server process model architecture scheme complies with such requirements. It can be easily applied by the QNX synchronous messaging scheme. The proposed control application process architecture is structured as shown in Fig. 9. The processes are designed to perform the following tasks:

- **hardware server** performs access to the I/O board facilities and thus provides signal interface.
- **path planner, setpoint buffer and XY controller** serve for motion control tasks.
- **control logic** runs logical control procedures which interact with servo drivers and the external PLC.
- **data server** allows to other processes the use of common data that are stored in the shared memory area.
- **TCP/IP client/server** deal with the Ethernet network communication link.
- **serial comm. sender/receiver** communicate with the external PLC.
- **user interface** implements user input/output and G-code program interpreter.

1) The **hardware server** applies the QNX send-receive-reply message passing protocol. Access to the I/O board facilities is thus performed by sending messages to the server process. In order to prevent a priority inversion situation occurring the hardware server has a floating priority. It receives messages from other processes requesting a service or information and replies with an indication that the service has been completed. The priority of these messages floats to the priority of the highest priority process, which has a message waiting in the server's queue to be processed. The message queue of the server is priority driven. Messages are serviced in the order of the sending process priority: the highest priority first.

2) The **XY controller** implements the position controllers for two servo axes. Signal conditioning is triggered by the hardware signal on the interface board at the specified control rate. Hardware watchdog management is also covered in this module. Moreover, basic enable/disable functionality, handling of fatal error (such as following error), software & hardware limits, that are included in the module ensure safe, reliable and robust operation. Homing routine is also implemented in the process, since servo drives with incremental encoders must be initialized prior to regular operation. In order to achieve accurate cyclical processing of the algorithms, the programmable interval timer periodically generates interrupt request as well as the synchronization signal in order to latch signals' values and to inform the PC that new data is available. The interrupt is serviced by the interrupt handler, which triggers a short message and returns. In further execution the short message signal unblocks the controller process, which is in a blocked state. Since its priority is the highest among the all application processes, it pre-empts others and is scheduled to run without any delay. The controller priority level is also above that of the file system. This allows access to the disk drive without causing controller glitches.

3) The **path planner process and the setpoint buffer server** are tightly coupled: the path planner process generates setpoints on the basis of user path requests and passes them to the setpoint buffer. The path planner process sends the setpoints to the
buffer in bursts. After sending a burst of setpoints, the path planner will wait for a predefined period before sending the next burst of setpoints. The setpoint buffer is a FIFO buffer. Setpoints sent by the path planner are stored to be read out by the controller process in the order in which they arrived.

4) The control logic process operates as a software PLC: it performs logical control procedures. Additionally, it monitors the operation of the external PLC. Thus, synchronization between the CNC controller and the PLC is achieved. In order to achieve periodic data handling, a software timer triggers a short message, which unblocks the blocked control logic process. The high priority level, which is set lower than the controller priority but higher than others, allows this process to run and consequently satisfies the soft real-time requirements. However, motion control algorithms pre-empt the control logic process accordingly due to the nature of the performed tasks.

Remark: All the processes mentioned above in 1)-4), which provide low-level control over the machine, use the FIFO scheduling method. The controller and the control logic are time-driven processes. The others are driven by events. Thus, with the use of the reasonable assigned priority levels and the synchronized messaging scheme determinism of the process execution order is guaranteed. The controller is highly prioritized over the other processes, and the path planner pre-empts control logic because setpoints must always be available to the controller when needed.

5) The data server sets up and provides access for other processes to a shared memory area. Typically common data is shared between the data server and the controller process. The data server allows other processes to access the data stored in this shared memory area without encumbering the controller process. The floating priority prevents the priority inversion problem.

6) The user interface process encompasses a variety of submodules, which have assigned tasks such as system start-up, G-code interpreter, system monitoring, setting of the machine operation, user input/output, programming interface, and the information link. The user interface also provides full information and operation support for the GUI. The G-code interpreter issues path requests for the path planner and/or control logic commands. In normal operation the user input/output is implemented via the Ethernet and a dedicated interface, i.e. the application protocol is stacked on the TCP/IP. The user interface priority is set below the path planner but above the default priority processes spawned by a shell. Consequently, the path planner pre-empts the G-code interpreter that can be invoked again when the path planner is idle.

7) The TCP/IP server and client processes provide TCP/IP support developed on the basis of Berkeley UNIX 4.3 BSD stack. A dedicated application layer protocol has been developed that essentially presents the command interface. The protocol states the meaning of the message. The information scheme is to be as tiny as possible to achieve a high refresh rate on the GUI side. Therefore, the cyclically refreshed information block includes only a general machine status. However, many other sporadic events must be automatically processed within a short response time. Consequently, they are supported by the special event-driven messaging scheme. The messages are sent when events occur.

8) The ser.com. sender and receiver processes provide data exchange between the external PLC and CNC controller on the basis of half-duplex serial communication. The PLC supplies the requested data that consists of variables with a predefined meaning. Each variable type is distinguished by an identifier. The variables which have been received are preprocessed and saved to the shared memory. Thus, they are available to the other processes.

Remark: The processes mentioned above in 5)-8) use the adaptive scheduling method. The processes with same priority level share the CPU to achieve best performance.

V. RESULTS

The CNC controller has been implemented on a PC with AMD ATHLON 800MHz Thunderbird processor on GA-7IXE4 motherboard with 64MB RAM, connected by Fast Ethernet (100Mbps) link to the GUI computer and RS232 connection with external PLC. The communication bandwidth is set by 19200 Baud transfer rate that assures the communication period of 200ms. The position control period achieved by the proposed system ranges from 200us, typically 1ms. The interrupt and the schedule latency are negligible compared to the desired control period. The control logic process was executed at 4ms period. The GUI data was cyclically refreshed at 0.5s periods.

The position control algorithm was of PI-P type. The PI part was coded in the CNC computer, the P part was involved in the servo drivers. Consequently, the output of the CNC position controller was command speed signal. However, the position control algorithm was finally executed at 250us time intervals. The position control results are shown by Fig. 10, Fig. 11, and Fig. 12. Fig. 10 present the test contour in the XY plane. The dotted line and the solid line denote the desired and actual curve. Fig. 11, and Fig. 12 present the trajectories of the control experiment in each axis. They include the reference speed, the position tracking error, and the control output signal (that commands the servo drivers). The results show high performance position tracking.

VI. CONCLUSION

The paper presents the application of CNC controller using off-the-shelf hardware PC with Ethernet network boards and commercial QNX RTOS. Two computer architecture networked by Ethernet TCP/IP have been suggested in order to add the HMI component of the control system. This kind of application is especially suitable for those small industries that deal with rapidly changing production lines, small repetitivity, and limited production volumes.
Openness of a control system is important issue in modern distributed automation environment. The QNX has opened recently; its source code is now available. Furthermore, it is open from the development point of view: only the kernel and other core portions of the QNX platform are protected, the modular architecture allows for extension of the system without any kernel coding. Hence, the proposed CNC software structure is open, since it has been built modularly; new features or number of control axes can be added easily thanks to the use of the QNX flexible IPC messaging scheme. In this paper the argument that switched Ethernet with TCP/UDP/IP stack can be applied for real-time industrial networks has been presented. It has also been shown that Ethernet and TCP/IP can be used for the implementation of the controller link with Microsoft Windows based HMI, however, interoperability that is required for a market-wide use can be upgraded by an open-standard application layer protocol. OPC [17] can be a good candidate for linking with Microsoft Windows platform, which is widely used in the factory environment. Moreover, recent developments in the area of object-oriented design, client-server computation, the networking technologies, and the creation of CORBA (Common Object Request Broker Architecture) can also allow for distributed software interoperability [16]. CORBA compliant products for QNX are now available and applied in the field of robotics [4].

The control system for the waterjet cutting machine reveals how to use modern technologies to build a control system. The presented approach can be used in solving the open control architecture problem. The application serves as a paradigm modular structure, which divides the system on separated functioning subsystems that are interconnected by a network. The system development time is also an important issue. Good technical documentation and support of the commercially available real-time operating system has helped developing control software and minimizing the development costs significantly. Finally, it may be pointed out that the application involves very potential technology with expected high grow in the field of factory automation.

APPENDIX

This appendix provides translation of the GUI windows depicted in Fig. 8.

The **procedure window**:

i. title: "TRANSPORT PHASE"

ii. right buttons:

- "the procedure stopped"
- "end the procedure"

iii. window list:

- "emergency stop released"
- "safety doors closed"
- "machine mains on"
- "pneumatic supply on"
- "pneumatic system ready"
- "supply ready"
- "transport system ready"
- "transport system drives ready"
- "safety barrier front ok"
- "safety barrier rear ok"
- "button pressed"
The faults:
"Errors:
[5]: X-axis servodriver error
[6]: Y-axis servodriver error
[37]: Wrong operation mode"

The message history:
"12:32:07[2292]: safety doors closed
12:32:07[2360]: PLC error
12:32:07[2131]: procedure stop
12:32:12[2842]: press button
12:32:12[2841]: press start button"

REFERENCES


Aleš Hace received the B.S., M.S. and Ph.D. degrees in electrical engineering from the University of Maribor, Slovenia, in 1994, 1998, and 2001, respectively. From 1994 he has worked at the Institute of Robotics, University of Maribor, where he currently serves as a senior researcher. He was a Visiting Research Fellow at Mechatronics Laboratory, Dept. of Mechanical Engineering, University of Loughborough, UK, in 1999. The main areas of his research are mechatronics, robotics, servo drives and control. The main research interest is motion control and advanced control techniques with the application on machine tools.

Prof. Karel Jezernik received B.S. (1968), M.S. (1974) and Dr.Eng. (1976) degrees in electrical engineering from the University of Ljubljana. He was a Visiting Research Fellow in Institute of Control, TU Braunschweig (1974-1975). In 1976 he joined the University of Maribor and in 1985 he became a Full Professor and Head of the Institute of Robotics. His research and teaching interests include automatic control, robotics, power electronics and electrical drives. Current projects in these areas are high precision tracking control in machine tools and DD robots and robust torque control in EV's. He consults in industrial servo control systems and other control and computer applications. Dr. Jezernik is an active member of the IEEE IES.