Int J Adv Manuf Technol (2008) 39:291–301
DOI 10.1007/s00170-007-1216-4

ORIGINAL ARTICLE

A software oriented CNC system based on Linux/RTLinux

Hua Ji · Yan Li · Jian Wang

Received: 7 March 2007 / Accepted: 20 August 2007 / Published online: 28 September 2007

Abstract The software-oriented CNC is one of the ideal solutions for open architecture CNC. However, its implementation is a difficult task, especially the guarantee of system's real-time performance. In this paper, a software oriented CNC system named Lin-soft CNC, whose operating system (OS) is Linux with its real-time extension - RTLinux - was proposed and depicted. The proposed system consists of four layers: the GUI, non-real-time layer, real-time layer and driver layer. Both the system and subsystem layout are detailed, along with the relevant implementation detail, such as the hierarchy design and the data communication between layers. Furthermore, the real-time layer has been identified as the kernel of the system, in which a simple and effective strategy - rational design of data buffer and high precision period of the real-time thread - is adopted to guarantee the real-time performance. At last, the prototype controller and test results are present.

Keywords Linux/RTLinux · Output-consumption chain · Real-time · Software oriented CNC

H. Ji · Y. Li (✉)
School of Manufacturing Science and Engineering,
Sichuan University,
Chengdu 610065, People's Republic of China
e-mail: liyan@scu.edu.cn

J. Wang
School of Mechanical and Aerospace Engineering,
Queen's University Belfast,
Stranmillis Road, Belfast,
Northern Ireland BT9 5AH, Northern Ireland

1 Introduction

Open architecture control (OAC) is a well known terminology in the field of machine control. In a similar fashion to the manner in which openness has revolutionised the PC industry, it also has the potential to revolutionise the CNC industry [1].

The open architecture (or open system) is not a new concept in the field of software engineering, with the IEEE 1003.0 providing a clear foundation of the definition of the open system in 1990. However, a common agreed definition for OAC still has not been reached in the academic world [2, 3], though many research groups, such as OMAC in U.S., OSACA in E.U. and JOP in Japan, have respectively expressed their viewpoint about OAC.

If openness is considered, the control systems can be categorised as open HMI, kernel with restricted openness and open control system [4, 5]. Open control systems that are available today normally offer the possibility for modifications in the non-real-time part in a fixed software topology.

PC front end, motion control card with PC and software-oriented CNC systems are the three main kinds of OAC in the current market, each with a different level of openness [6]. The PC front end is a traditional CNC black box with a personal computer attached to it, such as FANUC Series 150/160/180/210. The personal computer is incorporated in order to improve the interface for operator, although this kind of CNC is not an open control system in the strict sense. Motion control card with PC is configured by rigging an NC function board in a commercially available PC. The extra card, usually DSP-based, performs the time-critical NC kernel tasks, while the PC is for non-real-time functionality. The two CPUs communicate with each other

either by PC-bus or dual port RAM. This scheme improves the interface and offers flexibility to the machine-tool builders and end users.

The software-oriented CNC is an ideal solution to meet the real "openness" requirements of both today's and future markets [4, 7, 8]. With the drastic advancement of PC hardware and software technology, the time-critical CNC kernel task can be implemented by software running on PC with the support of a real-time software environment, usually a common operating system with a real-time extension. Open architecture control then transforms from a device-oriented to a software-oriented system.

The main idea of software-oriented CNC is making the most of the software to fulfil the function of CNC, integrating off-the-shelf hardware and software technology in order to achieve flexibility, agility and upgradeability. The interpolation, loop closing and PLC, as well as other real-time machining controls, are all performed in software on a PC. It reduces control and life-cycle costs, and increases machine tool uptime and productivity. In the software-oriented CNC system, the control systems can also be easily upgraded with the advances of software technology, while providing a simple and common methodology for communication and networking technology. Rapid advent of the embedded technology in the CNC systems, of which the software is the soul, can extend the service life of software oriented CNC. However, the openness and upgradeability in a controlled manner is a pair contradiction, which has to be managed carefully. The responsibility of the performance of the system after upgraded should be made clear in practice.

The numbers of software implementation methods of OAC are just as various as the definitions of OAC are. However, available OAC products on the market and OAC research work are almost exclusively based on Windows or Windows with real-time extension, such as MDSI Open CNC, Fanuc 210i/210is, Allen Bradley 9/PC or Siemens E&A 840D/840Di [2, 4], although DOS has also been used in the early products.

Although the research work of CNC based on Linux is still not as active as the equivalent Windows-based research [3, 8–15], the software-oriented CNC base on Linux/RTLinux is an open source software (OSS) solution, which has many advantages that Windows system cannot, and will not, offer. OSS is becoming a valuable part of the manufacturing system [16, 17] in which the users, integrators, and even developers can benefit through the flexibility of decision making, including which features to include, what extensions to make, and when to implement them. The large quantity of software offered by the worldwide open source communities avoids the potential for repetition of research work. Extendibility, scalability, upgrade ability and interoperability, which are the characteristic of OAC, can be implemented easily by adopting this software development pattern. OSS solutions also reduce the total cost of ownership by utilising software packages from the worldwide open sources.

Besides the common benefits of OSS, Linux, as an operating system, provides numerous benefits for the software-oriented CNC as regards the technical attributes, amongst which are:

- Robust, reliable memory-protected architecture;
- Fault isolation/management, limiting fatal errors to single processes;
- Availability of communication protocols, tools, and device-driver support;
- Support for key CPU architectures;
- Various options for OS and tools;
- Standard application programming interfaces (APIs) to foster software reusability; and
- Easy refitment to embedded systems.

Although available OAC products based on Linux are scarce on the market, the development of work within this field has never ceased. The frames of CNC based on Linux with real-time extension are discussed in [18–22], but the implementation of software-oriented CNC in the new environment - Linux - is hardly considered. EMC [23], an open source project by NIST, is a software-oriented CNC running under Linux. Originally, RCSLib (the real time control system architecture), a common real-time function library, was implemented in the system in order to assure the real-time performance and to provide the communication tool. The application of EMC1 proves that this strategy is complex and ineffective. Therefore, EMC2 was released in 22 May 2006, utilising a different strategy in which RTAI and RTLinux were used as Linux's real-time extension. However, in this software architecture, the communication method is the point within the system at which the problems start to arise. In EMC2, the SHM (shared memories) are used as the data buffers for the communication between Linux processes and real-time threads, and also for the communication between the real-time modules. To avoid the conflict of the threads accessing the same shared memory simultaneously, the mutexes are used every time when the shared memory is read or written. The mutex's over-usage lowers the efficiency and increases the complexity of the system. In hal-layer, the linked lists serve for the data structure of the data buffers, and exist in the shared memories. The CNC's data flow is mainly sequential: a module consumes the data produced by previous module, and outputs the data for the next modules. The manipulation of the linked lists in EMC2 is only getting data at the beginning then appending resultant data at the tail. However, real-time FIFO (RTFIFO) may be more suitable for the data flow of CNC and easier to

manipulate than the shared memory in the CNC system, which is implemented in this research.

In short, the software-oriented CNC is one of the ideal solutions for open architecture CNC. OSS also offers many benefits for a software oriented CNC. RTFIFO is suitable for the implementation of the data buffer, which sheds light on solving data flow and communication problems in the existing OAC systems based on Linux/RTLinux. Therefore, a software-oriented CNC-Lin-soft CNC - is proposed in this paper. Lin-soft CNC can benefit from both the software-oriented and OSS solutions. The construction of system platform in Linux, such as the selection of real-time extension and the development tools, is discussed. Four layers are used to descript the system hierarchy, which is helpful for the system's extension and the software development. The main data flow of CNCs is the output-consumption relationship. A simple and effective mechanism to maintain the output-consumption chain and guarantee the real-time performance of the system is presented. RTFIFO is selected as the implementation of the data buffer in Lin-soft CNC. In addition, Linux is only a soft real-time OS. CNC kernel tasks, such as interpolating and position controlling, etc., need hard real-time OS to support for time-stringent responses. To meet the hard real-time requirement of CNC, Linux needs a real-time extension. RTLinux is adopted as the real-time extension in Lin-soft CNC because it is one of the successful extensions, and we are more familiar with this system and have more references and experiences.

2 The real-time extension of OS in Lin-soft CNC

Windows and Linux are the two main operating systems which run on PC. As they are soft real-time OS, they both need the real-time extensions to meet the real-time requirement of CNC. VenturCom's RTX, Radisys's Intime, and Imagination System's Hyperkernel represent some of the real-time extensions for Windows.

For Linux, there are two approaches to facilitate the hard real-time performance, known as "preemption improvement" and "interrupt abstraction" respectively [24]. The first approach is implemented in TimeSys. TimeSys Linux modifies the standard Linux kernel to make it fully preemptable. RTLinux uses the second approach, which adds a virtual hardware layer between the standard Linux kernel and the computer hardware. By this way, RTLinux implements a complete and predictable hard real-time OS with no interference from the soft real-time Linux. Therefore, RTLinux provides an ideal environment in which a complete general-purpose operating system runs on top of the small hard real-time OS.

RTLinux is also an open source OS extension, and is widely used in commercial, industrial, academic and scientific development environments [25]. In the worst scenario, the interval between the moment a hardware interrupt is detected by the processor and the moment an interrupt handler starts to execute, is under 15 microseconds for RTLinux running on a generic x86. The RTLinux periodic task runs within 35 μs of its scheduled time on the same hardware [26]. Although there are still some flaws when using Linux with RTLinux as OS [27], these can be overcome by the careful design of the software.

The operating system plays an indispensable role in the software-oriented CNC system, and using Linux with RTLinux as the OS can meet the necessary requirements. First, the excellent real-time performance of RTLinux can meet the hard real-time requirements for software-oriented CNC. Secondly, Linux can meet the OAC's routine requirements [7] for access to TCP/IP, graphical display systems, file and database systems, and other services that are neither primitive nor simple.

So the Lin-soft CNC uses Linux with its real-time extension - RTLinux - as OS, as shown in Fig. 1. Off-the-shelf PC hardware is employed by Lin-soft CNC, in which a multifunctional card is used to output a pulse sequence to the digital AC servo and provides I/O signals.

3 The development tools of system under Linux/RTLinux

Linux has always provided a rich programming environment. There are several different kinds of high-level languages running in the Linux environment, which include compiled languages, interpreted languages and P-code languages. C is a compiled language that tends to give excellent performance and has the fullest access to the OS. RTLinux is mainly developed using C, but it is difficult and inefficient to use C for the development of the GUI. Tcl/

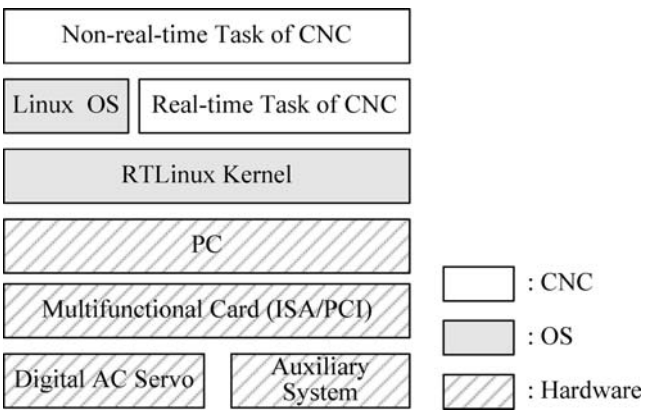


Fig. 1 The system architecture

TK, on the other hand, is an interpreted language that tends to be slower than compiled languages, and often has limited access to the underlying operating system and hardware. Nevertheless it is easier to program, and more forgiving of coding errors than compiled languages [28] because the interpreter prevents the developer from the high risk manipulations, such as the incorrect use of pointer. For these reasons, C is chosen as the development tool for the kernel code of Lin-soft CNC, while Tcl/TK can be used to develop the graphical interface for C programs due to its ease in developing graphical interfaces and interaction with C to perform real-time tasks. Other development tools are also used, such as OpenGL for machining simulation, and shell for the software integration.

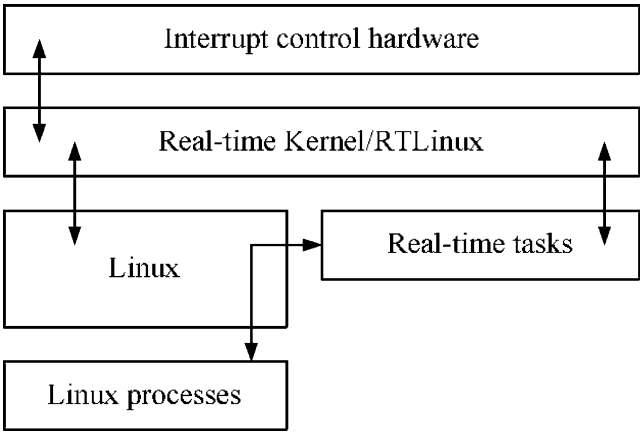


Fig. 3 Flow of data and control of Linux with RTLinux

4 The software hierarchy of Lin-soft CNC

The Lin-soft CNC is developed as a complete software system with certain complexity, and should realise various machining purposes. A rational software hierarchy is helpful for the system development, openness and stability.

Based on the analysis of the data relations, the general functions, time-bounded requirements, and the characteristic of RTLinux/Linux, four layers are proposed to describe the software system - the GUI layer, non-real-time layer, real-time layer and driver layer, shown in Fig. 2.

The main functions of CNC are implemented in both the real-time and non-real-time layers. RTLinux implements a hard real-time OS and runs Linux as its lowest task, as shown in Fig. 3. In this OS architecture, the existing format and the running space of real-time and non-real-time tasks are different. Real-time tasks are compiled as modules and run in RTLinux kernel space. Non-real-time tasks are compiled as executive and run in the Linux user space. However, both real-time and non-real-time tasks are essential in the software oriented CNC for integrality and optimisation of the entire CNC system. Only those time-critical tasks of the CNC should be designed as the real-time tasks because not all tasks that Lin-soft CNC is performing can be completed in RTLinux kernel space. For example, the interaction with the user requires tasks to be executed in Linux environment. The data flow from the hardware is transferred from real-time tasks to non-real-time tasks, and completion of the non-real-time tasks refreshes the GUI for users' information; then, the user sends instructions through the GUI according to the information. Those instructions are processed as the non-real-time tasks, and then the real-time commands formed by the processes are transferred to real-time tasks in RTLinux

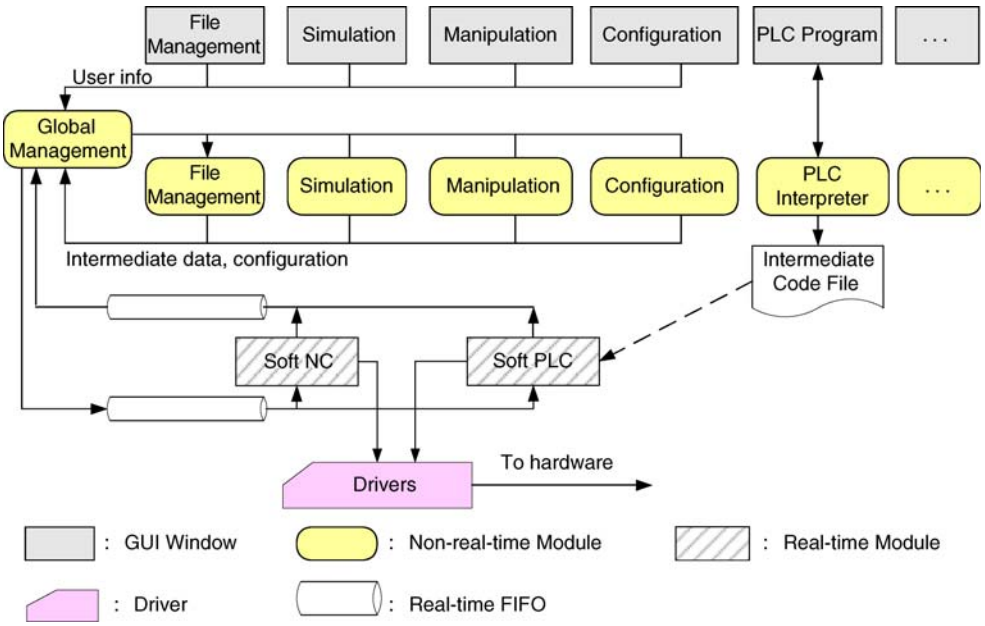


Fig. 2 The software hierarchy of Lin-soft CNC

kernel space. Much of the work that Linux can do, such as access to the Internet, cannot be completed by real-time tasks. In fact, the real-time tasks can only carry out real-time work. Therefore, non-real-time tasks must be designed in Lin-soft CNC. Additionally, the real-time requirements of CNC tasks are various, and not all tasks need to be running as hard real-time tasks. If all tasks are implemented in real-time, the whole system would be corrupted. Therefore, the real-time layer and the non-real-time layer are designed in the software hierarchy intentionally.

4.1 GUI layer

The GUI layer is only a graphical human-machine interface, which is responsible for accepting the instructions of the users' manipulation and displaying the machining information. Tasks like these are not time-critical. Thus, it is developed by Tcl/TK, which is an interpreted language and is slower than compiled languages. The GUI layer is not accountable for the real-time or time-consuming data processing, and is not designed for that purpose. However, this design methodology simplifies software hierarchy and the system development. In addition, it provides a more user-friendly interface with less effort.

4.2 Non-real-time layer

The non-real-time layer is the hub of the software, which is responsible for dispatching the data, pre-processing the data, triggering the real-time layer to commence the real-time task and translate the user's instructions into the real time commands. The modules in this layer, such as file management, NC code interpreter, PLC code interpreter, configuration and cutter compensation, run in Linux environment and are of non-real-time. The global management module runs in a loop firstly, and then waits for the user's instruction after the Lin-soft CNC system is switched on. It receives the user's instruction through the GUI, and then calls the relevant modules to pre-process the data. If the received instructions involve a real-time task, it completes the relevant process, and then passes the job and prepared real time commands to the real-time layer. It also receives the feedback data from the real-time layer and sends the data to the relevant modules, or presents it in the GUI.

4.3 Real-time layer

The real-time layer is the kernel of the system composed of modules, which run in the RTLinux environment. The real-time layer includes two modules: softNC and softPLC. SoftNC is accountable for the motion control and consists of interpolation, position control and other sub-modules. SoftPLC is designed for the logic control. When softNC and softPLC receive the control flow and data flow from global management module in non-real-time layer, they are activated and then run periodically in small time slice.

4.4 Driver layer

The driver layer is separated literally, although the drivers are actually not different from other RTLinux programs. They abstract all hardware or device dependent parts by defining a set of data structures, macros and functions and provide standard API. By this way, the modules in other layers are not influenced by the hardware changes.

4.5 Communication between layers

As described above, the system is divided into 4 layers, the GUI layer is implemented with Tcl/TK, non-real-time layer runs in the Linux environment, and other layers run in the RTLinux environment. Therefore effective communications between layers are essential for them to cooperate with each other, and the communication methods between different layers are implemented by specific manners according to the development tool and running environment.

The communicated information between GUI and non-real-time layer are users' instructions from GUI layer to non-real-time layer, machining process events, and feedback from non-real-time layer to display on graphic interface. The first communication task is implemented with Tcl procedure commands. Tcl procedure commands are purely used to transfer data, and are followed by data processing within the management layer. The second communication task is implemented using the extended C functions in Tcl/TK library.

The communicated information between non-real-time and real-time layer includes: 1) the commands sent from non-real-time layer to real-time layer, such as start command of single axis, G-code loop, softPLC, etc., 2) the corresponding control data to the particular commands must be sent before those commands, 3) the feedback data from real-time layer to non-real-time layer, such as the axes' kinematics data, errors and etc. In fact, the communication happens between real-time task and non- real-time task, i.e. RTLinux threads and Linux processes. Two methods, real-time FIFO and shared memory, are provided by RTLinux to fulfil this task. The real-time FIFO is simpler and more flexible. Moreover, synchronous IO means is also provided, and it can save communication time [29]. Furthermore, the RTFIFO has been selected to simplify the communication manner in the system because the RTFIFO is also suitable for the communication of the

modules in the real-time layer. So the communication is implemented by real-time FIFO.

5 Real-time layer in Lin-soft CNC

The real-time layer is the kernel of Lin-soft CNC and runs in RTLinux environment, which includes softPLC, softNC. The two modules collaborate to fulfil the machining tasks.

5.1 SoftNC

As shown in Fig. 4, SoftNC includes four sub-modules: pre-process, interpolation, position control and output. Each sub-module runs in RTLinux environment as a real-time thread. The RTFIFOs act as the data buffers. First, the output thread commences running when the global management module in non-real-time layer triggers the softNC. Those threads run in the pipelining manner, writing their own real-time FIFOs and reading from relevant real-time FIFOs. Finally, the Output thread outputs signals via drivers.

In normal CNC system, the so-called "position control" actually includes the position control and output which are combined into a sequential action, outputting to hardware directly after the position control. As the time sequences of these two tasks are not independent of each other and the time consumption of each output is different, the accurate time sequence of the position control cannot assure the accuracy of the output. So the position control and output are separated in Lin-soft CNC, running as two threads so that both time sequences can be guaranteed strictly, especially for the output.

Each sub-module is compiled as a "real-time module" which exists in the file system as a *.o. This encapsulation makes the module with independent function easy to replace. Therefore, the independence of module and the openness of the system are enhanced.

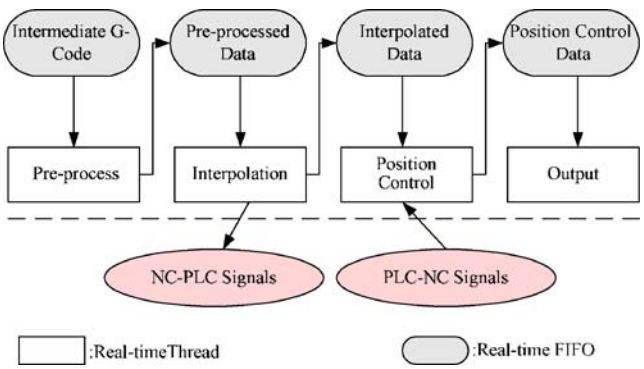


Fig. 4 The work principle of SoftNC

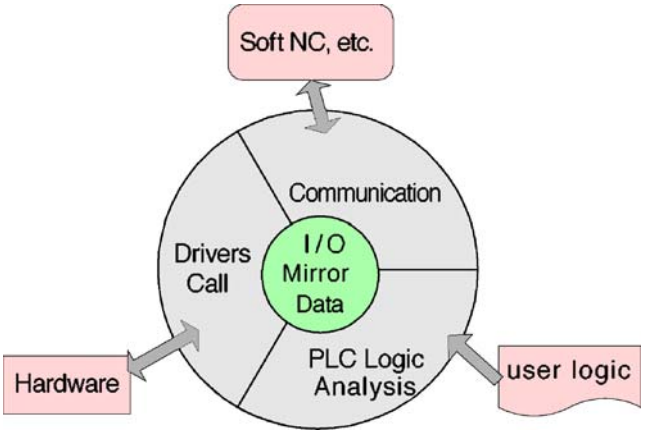


Fig. 5 The working principle of softPLC

5.2 SoftPLC

Intermediate code of user's logic is generated and saved in a file after PLC programming, including code writing, error checking, simulation etc. SoftPLC loads the user's logic from the fixed file when it initializes.

The I/O mirror data table is the kernel of the softPLC, which includes: 1) NC-PLC datasheet; 2) PLC-NC datasheet; 3) machine-PLC datasheet; 4) PLC-machine datasheet. The softPLC is executed by a scanning method and runs as a real-time thread. It calls the driver to input the machine-PLC signals, analyses the user's logic based on the machine-PLC datasheet and NC-PLC datasheet, writes PLC-machine datasheet and PLC-NC datasheet, and then outputs to hardware. It also has the functions to read NC-PLC datasheet and writes PLC-NC datasheet in the thread.

The working principle of softPLC is shown in Fig. 5.

5.3 Communication between softNC and softPLC

When the system starts, the softPLC runs in a loop. The softNC is triggered when the task arrives, and suspends after the task is completed. SoftPLC and softNC communicate through the shared memory, i.e. PLC-NC and NC-PLC datasheets. The instructions are written to PLC through the interpolator. If the interpolator picks up the instructions in pre-process data buffer, the interpolator locks the NC-PLC datasheet and interpolates the corresponding bit instantly, then writes them in the NC-PLC datasheet. When this task is completed, the interpolator continues to search for other instructions. In this way, it can be assured that the instruction can be done at the right moment in the correct way. The position control also reads the PLC-NC datasheet to query PLC-NC signals. It checks the head of PLC-NC datasheet regularly that contains the number of signals it carries. If this number is zero, the position control

returns to the other job. In the same manner, the softPLC communicates with softNC.

5.4 The strategy to guarantee real-time performance of the system

The real-time performance of Lin-soft CNC is ensured mainly by the high precision of timing period of RTLinux and the reasonable design and management of the data buffer. Both the period and the size of the data buffer can be adjusted by the requirement in Lin-soft CNC.

In the typical CNC's data flow, G-codes are executed in order. A G-code is executed in turn by the G-code interpreter, pre-process, interpolation, and position control module, then, the output module output to the drivers at the end. This is a typical first-in-first-out model for the data flow, and a typical output-consumer relation for the modules. Each module consumes the previous module's output data in sequence, and outputs data to next module in sequence. RTFIFO provided by RTLinux cannot only serve for the medium of the data buffer, but also is the first-in-first-out from the point of the data structure. So the RTFIFO is selected as the data buffer of CNC.

Each thread runs in a fixed period and with a fixed priority. The output thread runs with the highest priority. In a running period, each thread checks whether the data in its consuming data buffer is enough, and then computes and outputs data to its output data buffer if the data in its consuming data buffer is sufficient.

In normal situations, each thread runs without restriction from other threads. Each thread just consumes the consuming data buffer and outputs data for the next thread. The output-consumption relation (suffice of each data buffer) is maintained by the thread's priority and the computer's computation ability. Because the PC's computation is powerful nowadays, each thread can get its chance to execute normally.

In special situations, prioritisation has to be enforced in case there are enough data left in the data buffer. A simple and efficient strategy is used. For example, when Lin-soft CNC starts, the output thread, which has the highest priority, starts first. The output thread consumes the data in RTFIFO of the position control, and then activates the position control thread when it recognises that there is no data in the buffer. The position control thread outputs the required data, and then suspends itself when it fulfils the buffer. The interpolator also functions in this manner. The last half size of the data in the buffer is marked with a finishing sign, the output and position control thread process these data and then exit.

In the prototype system, the pre-process thread consumes one unit of data from the pervious data buffer, computes this unit of the data, and then writes the results into the pre-process data buffer in one thread period. However, the interpolator thread consumes the data from the pre-process data buffer, interpolates 25 times (half of data buffer size), and saves the 25 results into the interpolator data buffer in one thread period. In the same manner, the position control thread consumes 25 units of data from the interpolator data buffer, computes them respectively, and then produces 25 units of data in the position control data buffer in one thread period. The output thread consumes one unit of data from the position control data buffer, computes it, and then output to hardware in one thread period. This is because the time consumption of interpolator and position control thread is very small in each computation as shown in Table 1, and the switching between threads can consume too much time if each thread only calculates once in a period. Therefore, the position control thread recognises there is less than half of the buffer data left in the interpolator data buffer, it suspends itself. Other threads recognises there is no data left in previous data buffer, they suspend themselves.

This strategy of maintaining the output-consumption chain not only can avoid the complicated synchronization control of the threads, but also can effectively avoid the thread delay. The test proves that this strategy can guarantee the system's real-time performance.

Table 1 The calculation time and frequency of G-code

Thread	Line 1 of G-code					Line 2 of G-code				
	Calculation once, μs			Calculation frequency		Calculation once μs			Calculation frequency	
	Min.	Aver.	Max.	In one period	Total	Max.	Aver.	Min.	In one period	Total
Pre-process	1.47			1	1	2.23			1	1
Interpolator	0.64	0.70	0.99	25	126	13.40	13.82	18.88	25	105
Position control	0.16	0.18	0.23	25	126	0.16	0.20	0.26	25	105
Output	22.75	23.40	28.26	1	126	22.80	23.43	22.31	1	105

The G-code for test is:N01 G01 X0 Y100 Z0 F3000N02 G02 X50 Y50 R50

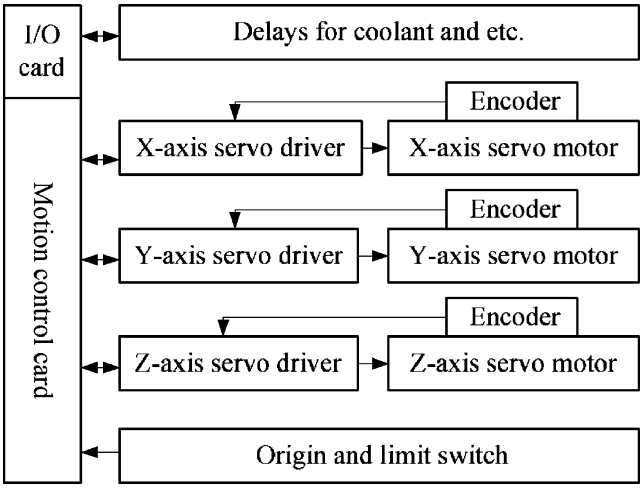


Fig. 6 The hardware of test bed

6 Application exemplar

6.1 The prototype controller

The prototype of an open CNC controller for a milling machine test bed was developed using the Lin-soft CNC. As shown in Fig. 6, the milling machine test bed consists of typical I/O and 3 AC servo-motors GYS101DC1-CB (FUJI, Japan) for the x-axis, y-axis, and z-axis. Every motor driver RYB101S3-VBC (FUJI, Japan) and I/O are connected to a common multi-function control card MPC04. (A FPGA based card, including the I/O card and Motion control card in Fig. 6, made by step-servo, Chengdu China), which is plugged into a general IPC (CPU: Intel Pentium(R) 4 CPU 2.00GHz; SDRAM: 512M).

Details of the kernel modules implemented in the exemplar are shown in Fig. 7. As mentioned above, Modular 1 runs a Linux Process. For Modular 2, 5, 6, 7 and 8, each of them runs as a real-time thread. Modular 3 and 4 in the same thread run as a real-time thread. Modular 9 provides the function call to manipulate hardware for the related modules.

The main work of pre-process is to generate tool paths from a set of decoded G-code. The main results are the velocity, start point and end point of tool paths. Both the linear interpolation and circular interpolation were implemented. The circular interpolation arithmetic is quadric recursion circular interpolation [30]. The position control in the prototype system mainly transfers the interpolated results into the control instructions because the motion control card is a pulse control card and the position feedback is connected to the servo driver, as shown in

Fig. 7 The kernel modules implemented in the exemplar

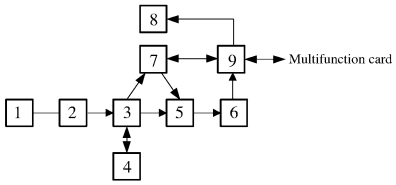
The software modules	Modular No.	Modular name	P	I	C	D	Description
	1	Decoder				1000	To decode G-code
	2	Pre-process	6	10	7	10	To generate tool paths from a decoded G-code
	3	Interpolation	7	5	6	50	To interpolate a tool path
	4	Acc/dec					To plan acceleration and deceleration
	5	Position Control	8	5	5	50	To generate the axis movement instructions
	6	Output	9	20	1		To output the generated instructions to servo drivers via calling the drivers
	7	SoftPLC	8	10	2	10	To generate the PLC control instructions
	8	Movement State	9	20	3	10	To generate the current position and the velocity via calling the drivers
	9	Drivers					To abstract the hardware and provide the API
The relations of software modules							
Notes:	P – Priority of real-time thread; the higher priority of real-time thread is represented by a bigger number. I – Running Interval (period) of real-time thread, ms C – Creating sequence of real-time thread D – Outputting data buffer size of the modular (One data is produced while a real time thread runs for once)						

Table 2 Test conditions and results

No.	Period of thread		Test result	
	Interpolation	Position control	GUI update	Output thread suspending because of lack of data
1	5 ms	5 ms	Normal	5%
2	1 ms	1 ms	Lagging	2%
3	0.2 ms	0.2 ms	Not	1%

Fig. 6. The position control thread does not deal with the position compensation, which is processed by the servo driver.

The period of interpolation is set to 5 ms at standard running condition in the system because of the nature of relatively time-consuming. The period of position control thread is also set to 5 ms because the period of position control is equal to or a multiple of the period of the interpolation thread.

The priority of output thread is set as the highest because hardware execution depends on output thread to provide data and the sequence of output thread must be guaranteed strictly. The period of output is set at 20 ms. This is because the period of output thread must match the period of servo driver that is longer. In addition, although the calculation time in output thread is small, the period of output can be long.

The size of data buffer outputted by decoder is set 1000, because the decoder is a Linux process and has lowest priority to be scheduled by RTLinux. If the CPU is idle for the decoder to run, the decoder does its work to output data to the buffer, preparing for the real-time threads. Bigger size of data buffer allows the decoder outputting more prepared data if the decoder has opportunities to execute. The data buffer size of the interpolator and output is bigger than that of the pre-process because the previous threads run more frequently.

6.2 Test and result

6.2.1 Test method

For the testing and analysing purpose, watch points, such as the point of the thread suspending, the thread entering, and the thread leaving etc., are set in the software. At every watch point, the rtl_printf function is employed for printing test-messages. The test-messages (kernel messages) printed by rtl_printf (like printk) can be outputted to a file by employing the Linux command - dmesg > testfile for analysing purposes. Therefore, a shell creating test result files starts before the Lin-soft CNC stards and continually runs when the Lin-soft CNC does. From the saved files, the running states of real-time threads, such as entering the thread, suspending the thread, and leaving the thread, etc., can be investigated freely offline. The timing sequence of the real-time threads can also be checked by analysis of the running state. Because the execution frequency of Linux shell is not high enough to collect every test-message, the investigation is in a random mode.

6.2.2 Calculation time of threads

For testing the calculation time of main threads in the prototype system, the RTLinux function, gethrtime, is

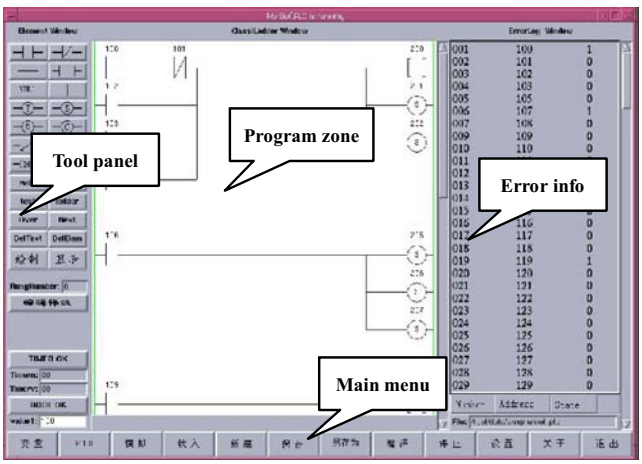


Fig. 8 HMI for PLC programming

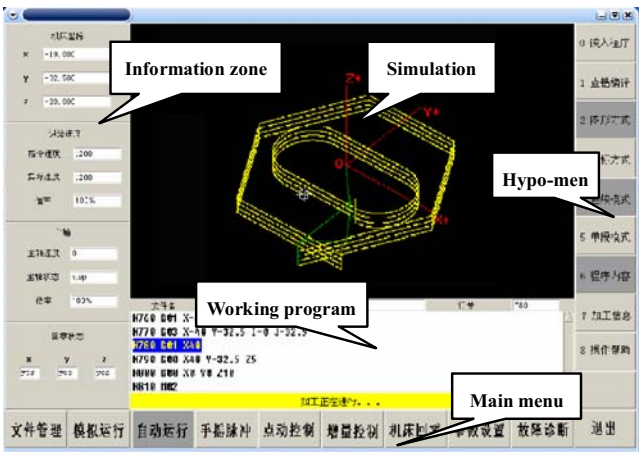


Fig. 9 HMI for the system running in automatic mode

added to the source code. The tested G-code shown in the Table 1 has two lines. The first line is designed for testing line interpolation, and the second one for circle interpolation. For the fist line, the pre-process thread works once and the thread of interpolator, position control, output work 126 times because every interpolated result transfers to pulses and outputs to the hardware. From the Table 1, it can be found that the calculation time is less than the period of the thread and the thread periods setting in Fig. 7 can satisfy the system's requirements

6.2.3 Influence of thread period on system performance

To test the software, three test conditions relevant to discussed scenarios are designed and shown in Table 2, in which the rest parameters are kept the same as in Fig. 6. The Lin-soft CNC performs well when the period thresholds of real-time threads are set as shown in Fig. 7. The GUI of Lin-soft CNC lags when the thresholds of position control and interpolation are reduced to 1 ms. In this scenario, the load of real-time threads becomes heavier. GUI process cannot be scheduled in time because it is a Linux process which runs as the lowest priority thread under RTLinux. When testing under extreme condition-scenario 3, the GUI is rarely updated or completely cannot be updated. The normal work of GUI is often necessary for a CNC system. When the GUI is scarcely updated, the system is worthless even if the real-time performance can be guaranteed. Therefore, the extreme condition can be defined as when the GUI of CNC cannot be updated in time.

Whether the suspending reason of the output thread is lack of data is an important criterion of the system performance in the multi-thread environment. The test result shows that 5% of runs suspend because of lack of data during 10^5 output runs for scenario 1, 2% for scenario 2, 1% for scenario 3, respectively. When time limits set for the position control and interpolation module are reduced, the real time performance of the system is enhanced, but the capability of the GUI cannot catch up. Therefore, in the extreme conditions (scenario 3) Linux process cannot satisfy the system requirements with a reasonable GUI. Because the speed of output thread is higher than the speed of the machine movement, 5% or even larger of suspending time can meet the system's requirements. From the analysis of the test results, it is also found that the time sequence in the system is correct. When there is not enough input data, the thread can suspend correctly.

Figures 8 and 9 show runtime cases of the system. Fig. 8 shows the HMI when the user programs with the ladder logic, while Fig. 9 indicates the HMI when the machining task is performed in automatic manner.

7 Conclusion

With the advent of PC hardware and software technology, the software-oriented CNC is becoming one trend of open architecture CNC used within todays and potentially future markets. This paper presents a software-oriented CNC system based on Linux/RTLinux, namely Lin-soft CNC, and discusses the architecture and the key technology in this new environment. The following can be concluded:

1) OSS solution is one realistic approach to open architecture CNC. Using Linux with RTLinux as its real-time extension can meet the requirements of the open architecture CNC's and OS, including the hard real-time requirements of CNC kernel tasks and the common OS services.
2) The data communication between layers acts as the key link between modules in the Lin-soft CNC. The communication between modules must be implemented by different techniques in Linux/RTLinux according to the differences of the communicating layers, such as the scripting language, the running space, etc.
3) The output-consumption relation is one key characteristic of data flow in the CNC system. The design of data buffer and high precision period real-time thread can maintain output-consumption chain and meet the real-time requirement of the system. The strategy maintaining the output-consumption relation in this paper is simple and effective, avoiding the thread delay effectively and complicated thread synchronization control. In addition, using RTFIFO as the implementation of the data buffers simplifies the communication of CNC system.
4) Although the test of prototype controller shows that the design can meet the real-time requirements of CNC system, there are still some future work, especially the optimisation of period, priority and data buffer size for better performance of the system.

Acknowledgements The authors would like to acknowledge the support of funding by the Key Technologies Research and Development Program of Sichuan Province, China (No. 03GG006-013-1, No. 2006Z01-012) and the Key Technologies Research and Development Program of Chengdu City, China (No. 06GGYB052GX).

References

1. Ambra C et al (2002) The design of a high performance modular CNC system architecture. In: Proceedings of the 2002 IEE international symposium on intelligent control, British Columbia, Canada, 27–30 October 2002, pp 290–296
2. Katz R, Min BK, Pasek Z (2000) Open architecture control technology trends. http://erc.engin.umich.edu/publications/PubFiles/TA3/OAC_Report_9_18.pdf

3. Wu HE et al (2006) Windows XP embedded based open architecture computer numerical control system. In: 2006 IEEE/ASME International Conference on Mechatronics and Embedded Systems and Applications, Beijing, China, 13–16 August 2006
4. Pritschow G et al (2001) Open controller architecture-past, present and future. Ann CIRP 50(2):463–470
5. OSACA (1996) OSACA I and II Final Report. http://www.osaca.org/_pdf/os2fin4.pdf
6. Chi YL et al (2003) Software CNC system. Modern Manuf 15:110–112
7. Zhang CR et al (2003) An USB-based software CNC system. J Mater Process Technol 139:286–290
8. Park S, Kim S-H, Cho H (2006) Kernel software for efficiently building, re-configuring, and distributing an open CNC controller. Int J Advanced Manuf Technol 27:788–796
9. Nick P et al (2007) Single controlled axis lathe mill. Int J Adv Manuf Technol 32:55–65
10. Bin L, Yun-fei Z, Xiao-qi T (2004) Research on open CNC system based on architecture/component software reuse technology. Comput Ind 55:73–85
11. Chen ZY, Guo W, Li CX (2006) Research and development on full distribution CNC system. In 2006 1st IEEE Conference on Industrial Electronics and Applications, Singapore, 24-26 May 2006
12. Wang YH, Hu J, Li Y (2003) Study on a reconfigurable model of an open CNC kernel. J Mater Process Technol 138(3):472–474
13. Huang XH, Wang XC, Zhang Q (2006) The Research of the CNC System of the Plane Milling Machine. Proc 6th World Congress on Intelligent Control and Automation, June 21 – 23, 2006, Dalian, China, pp 7978–7982
14. Zhou ZD et al (2004) The development of a fieldbus-based open-CNC system. Int J Adv Manuf Technol 27:507–513
15. Ferraz F, Coelho RT (2005) Data acquisition and monitoring in machine tools with CNC of open architecture using internet. Int J Adv Manuf Technol 26:90–97
16. Bartos FJ (2004) Is Linux at the gates of the factory? Control Eng 51(5):32–36
17. Dennis B (2004) Open up with open source. Control Eng 51(5):56
18. http://www.fidia.it/english/research_penguin_fr.htm
19. Li WG et al (2003) Frame of open numerical control system based on RT-Linux. J South China Univ Technol (Natural Science) 31 (10):28–31
20. Zhao WS et al (2004) Research on six-axis numerical control system for EDM. Comput Integr Manuf Syst 10(10):1263–1268
21. Chen YD et al (2003) Exploration of open architecture CNC system software based on RT-Linux. China Mech Eng 14 (16):1419–1422
22. Ferrarini L, Veber C, Fogliazza G (2003) Modeling, design and implementation of machining centers control functions with object-oriented techniques. In: IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003), Kobe, Japan, 20–24 July 2003, pp 1037–1042
23. http://www.linuxcnc.org
24. OCERA (2002) WP1 - RTOS state of the art analysis. http://www.ocera.org/archive/deliverables/ms1-month6/WP1/D1.1.pdf
25. FAMLabs. Hard real time case studies. http://www.fsmlabs.com/case-studies.html
26. V. Yodaiken The RTLinux manifesto. http://www.fsmlabs.com/images/stories/pdf/archive/rtmanifesto.pdf
27. Hace Ales, Jezernik (2004) Control system for the waterjet cutting machine. IEEE/ASME Transactions on Mechatronics, 9(4):627–635
28. Aznar G et al (2004) How do computer languages work. http://www.linux.com/howtos/Unix-and-Internet-Fundamentals-HOWTO/languages.shtml
29. Dougan C et al (2003) RTLinux POSIX API for IO on real-time FIFOs and shared memory. http://www.fsmlabs.com/images/stories/pdf/literature/posix_fifo_and_shared_memory.pdf
30. Wang JX, Zhang GX, Liu JC, Dong S (1999) Development of ultra precise lathe CNC system. China Mech Eng 10(11):1216–1219