Robotica (2004) volume 22, pp. 339–343. © 2004 Cambridge University Press
DOI: 10.1017/S0263574703005575 Printed in the United Kingdom

A chess-playing robot control system based on Windows NT+RTX

Jilin He, Ruqing Yang, Qunfei Zhao and Chunxiang Wang

Robots Research Institute, Shanghai Jiaotong University, Shanghai 200030, China

(Received in Final Form: September 3, 2003)

SUMMARY
This paper presents the control system of a chess-playing robot developed by the Research Institute of Robots at the Shanghai Jiaotong University. Thanks to the Windows NT operation system and the RTX (Real-Time eXtension), the whole system can achieve good real-time performance. The control system, which is supported by a standard PC hardware platform and a modularized structure of system software, is open-ended and easily expansible.

KEYWORDS: Robot controller; Real-time control; Chess-playing robot.

1. INTRODUCTION
The chess-playing robot is a combination of computing, artificial intelligence, automatic control, and optic-mechatronics that is composed of optical, mechanical and electrical technology. Our chess-playing robot is a service robot with a wide-range of functions and capacity changes of adaptation. It was designed from the general robot by means of external, hardware design and development of various entertainment software. In order to combine this robot with the Chinese traditional chess-playing art, all kinds of effective technologies available, such as the calculating and memorizing capability of a computer, logic reasoning and judging ability of artificial intelligence, and reliable optic-mechatronic technology, were applied, and an authentic chessboard and chessmen were employed. Our hardware design endowed the chess-playing robot with human characteristics and the ability to interact with people in an audio-visual way at a higher level. The development of various chess-playing software enabled the robot to play chess with people intelligently. Here, the word chess is used in a broad sense and different software can be designed to improve the adaptation of this chess-playing robot to different situations. Figure 1 is the flow chart of the chess-playing robot.

The study focuses on Chinese chess, and therefore this paper, confines all the discussions to Chinese chess. Its open-endedness, however, enables us to easily further develop the robot for other games. Of course, to achieve this purpose, the information extraction, target identification and especially the real-time motion control become the key points to realize high-level interaction between the machine and the player, and improve the self-learning ability of the machine.

2. SYSTEM ARCHITECTURE
It is well known that a robot control system is a real-time system dealing with multiple tasks executed simultaneously. In addition, the requirements on the motion control system are very strict in terms of real-time control, security, robustness, and so on.[1,2]

In order to meet our demands, the real-time environment of Windows NT plus RTX (see Figure 2) was applied in this research. The WIN32 subsystem takes charge of the processes and threads manager to make the control program run smoothly and efficiently.[1] The Real-Time subsystem, at the same time, is responsible for the real-time property of the motion control).[3] The manager layer and the execution layer are illustrated in Figure 2. The former consists mainly of WIN NT, its kernel and so on. The latter consists-of RTX-RTSS (Real-Time eXtension-Real Time SubSystem), its HAL (hardware abstract layer), etc. Of course, there is a communication layer between them to integrate the whole application.[4,5] The motor, its actuator, arm, grip, sensor and so on can also be seen on the chart. They are the hardware components and relevant tools to constitute this chess-playing robot.

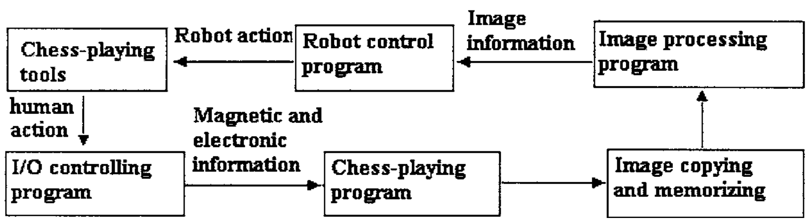

Fig. 1. Flow chart of the chess-playing robot.

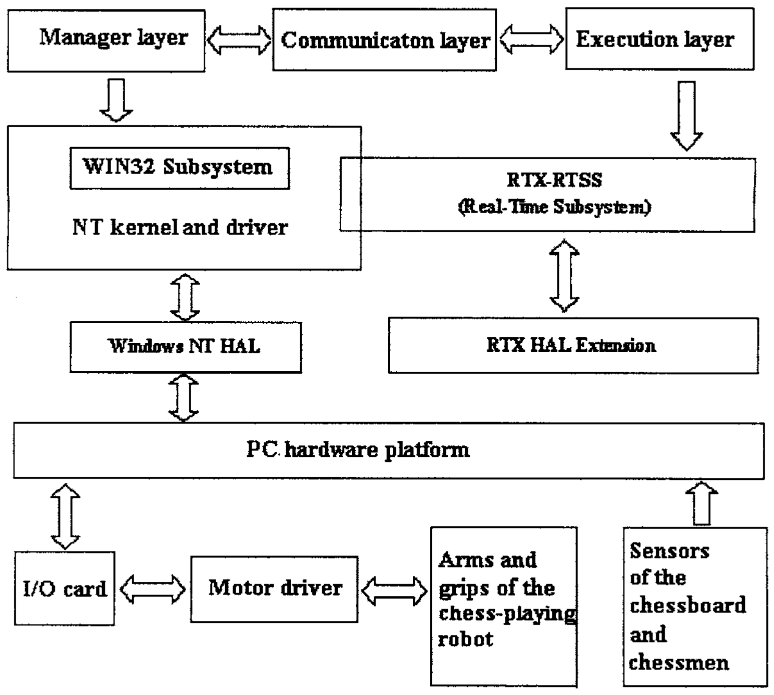

Fig. 2. Framework of the system.

3. HARDWARE ARCHITECTURE

The hardware architecture of the whole system adopts a modular structure (see Figure 3). The general IPC (Industrial Personal Computer) was chosen as the system hardware platform, with a high-quality I/O card controlling the motion of all joints.[6] The body of the chess-playing robot is a dual-arm humanoid robot with 6 degrees of freedom. With the expansibility of the IPC platform, the hardware module can be expanded if necessary. For example, an A/D converter and network card can be added to process external sensor information and deal with remote communication, respectively.

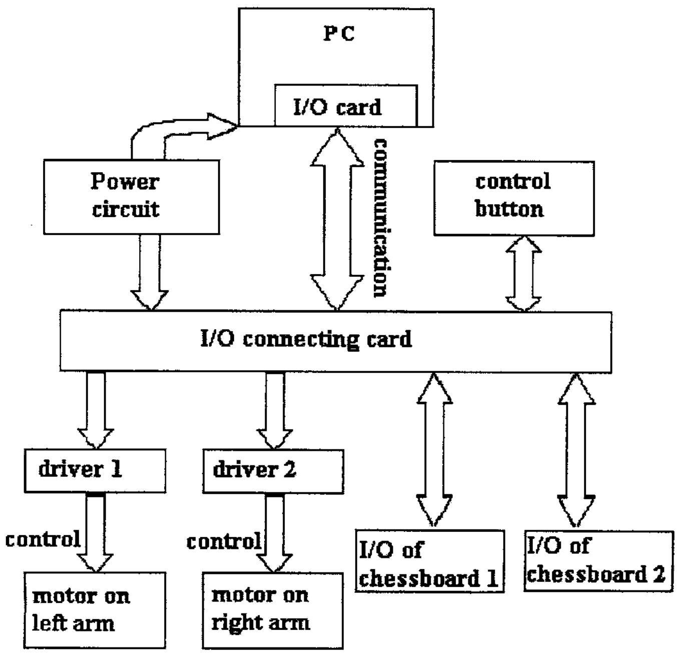

Fig. 3. Hardware architecture.

3.1. *Controller structure*

It is well known that a high-performance controller is the safeguard of an effector achievement for all control purposes.[7] The controller of our chess-playing robot includes a manager layer and a motion execution layer. Network devices, an IPC and a teaching box comprise the manager layer. With the network devices, the chess-playing robot can be controlled remotely and can be made into a network robot to some extent. The CPU and some other characteristics in the IPC make the whole robot system reliable in relatively bad environments. The teaching box is necessary for self-learning, specified point teaching, track teaching, etc.

The motion execution layer mainly consists of motors (AC servo or stepping) and their actuators that make the chess-playing robot's movement accurate and efficient. There are two groups of motor drivers, one of which is for the motors in the left arm, and the other for the motors in the right arm. In each group, different motors actuate the upper arm, the forearm and the grip to make them move independently.

3.2. *Peripheral hardware*

Two chessboards were made. Thus, the chess-playing robot can play Chinese chess with two different players. The working and measuring components monitor the operating status of the robot by means of various sensors, A/D and D/A, and provide feedback to guarantee the robot's security and reliability.

It is proved by practice that such hardware architecture takes advantage of every component's strong points and integrates all components excellently.

4. SOFTWARE ARCHITECTURE

The high-quality Windows NT plus RTX is chosen to safeguard the real-time property and so on.[1,3]

Table I. Differences in real-time properties between Windows NT and RTX.

	Timer accuracy	Resolution	Interrupt delay	Switch Time between threads	Process delay	Communication time/64 bits
WIN NT	10 ms/1 ms	0.84 us	25–7590 us	3–2885 us	45–100 us	50 us
RTX	100 us	0.1 us/1 us	8–14 us	10–20 us	8–20 us	5 us

Note: (Measuring condition: AdvanTech IPC PIII700).

4.1. Characteristics of WIN NT and RTX

Windows NT is one of the most popular operating systems. It is better in graphic user interfaces than in other systems and is supported by a wide-range of software and hardware. RTX further extends the real-time function of the NT system. Therefore, the abundant resources of NT and the real-time function of RTX are used in our research. Furthermore, the widely used debugging tools, such as VC++, WinDbg, etc, no doubt bring the powerful real-time system under our control.

The RTX real-time environment can fully achieve the process priorities of 0-127 and control the priority of the process to be executed.[3] It possesses an accurate clock with 1 μs resolution and a 100 μs accurate timer. What's more, the RTX system control is very efficient with a time delay of 45 μs in communication between RTSS process and WIN 32 process and of only 8 μs for inter-RTSS process communication (the outcome was obtained from an industrial PC of PIII700 and is listed in Table I). It is shown that RTX is better than NT in real-time situations and therefore guarantees higher quality pulses.

It is well known that in the NT system the execution programs can't operate the hardware directly. Fortunately, RTX gives us access to the hardware and provides us with a lot of real-time API (Application Program Interface), such as delay function with 0.1 μs resolution, port read/write function with high priority. RTX can, therefore, ensure the real-time property and accuracy of the entire control system.

4.2. Manager layer

The software architecture is divided into the manager layer, the task execution layer and the motion control layer (see Figure 4).

The operating and controlling console, the instructing and teaching module, an interface program between people and machine, the chess-playing program and remote network communication module are all in the managing layer.

The operating and controlling console is mainly for the interaction between players and the robot controller. The users can choose the diffculty level and the precedence relationship of playing so that they can improve their chess-playing skills step by step and gain a sense of achievement by choosing the right difficulty level and winning the game. In order to achieve a more vivid effect of chess-playing and better interaction between the robot and player, a real-time and on-line display of the whole playing process is provided. The display makes the game livelier and the robot self-learning. What's more, it can be converted into files if necessary.

The main function of the instructing and teaching module is off-line path planning, teaching and instructing. Several

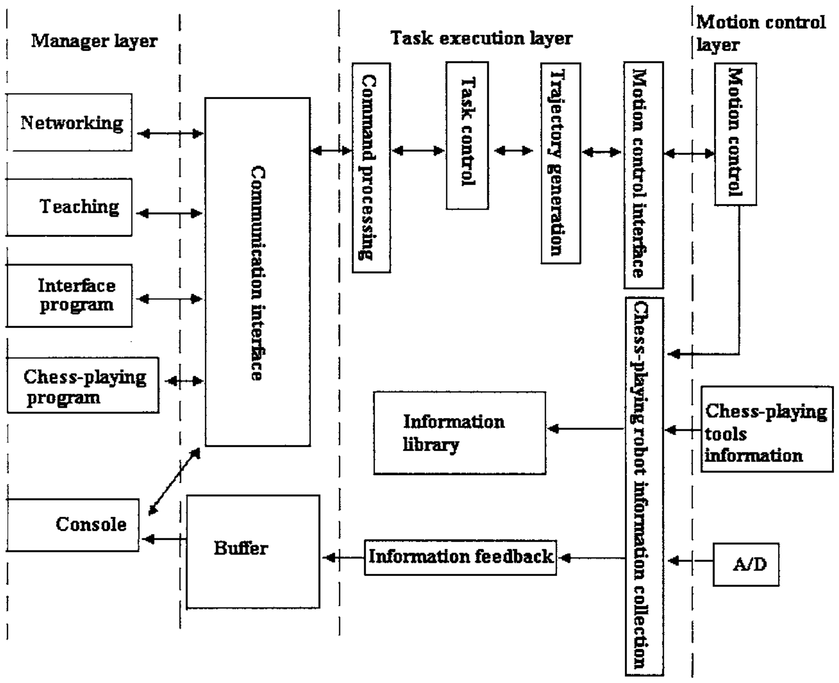

Fig. 4. Software architecture.

key points of the specified path were fed in the instruction box that can help modify the track interpolation of the robot arm. For example, the instruction box simulates the planned path by means of the stepping mode and modifies the position and pose whenever there is a difference. At the same time, the instruction box can be used as a monitor since it has a LCD (Liquid Crystal Display).

The interface program between people and machine facilitates interaction between the robot and the player, receives and processes the detailed information of the chess-playing tools based on data fusion and then impacts the responding tools.

The chess-playing software and the intelligent library running background in the IPC provide necessary conditions for the image-processing module to analyze and extract the information of the chess-playing tools. With the chess-playing program, the intelligent library and the relevant application program of image processing, the robot can make exact decisions and reach its goal.

As to the remote network communication module, the chess-playing robot is based on a standard IPC and therefore it is convenient to add other network devices. In addition, the rich net resources make it possible to control the robot from remote areas. And also many users can play the game at different places.

The traditional robot language is basically for a specific purpose and is consequently hard to be applied widely. The *VC* language is chosen for our purpose because it can provide a programmable interface for users to call. The position obtained by the instruction box is saved as position variable and then can be referred to in the chess-playing program. Furthermore, the chess-playing software was designed in *VC* by ourselves, so it is not only compatible but also enhances the real-time property of the whole system.

4.3. Task execution layer

The task execution layer gets instructions from the manager layer and forms the control instruction. There are command processing, task controlling, path generating and motion control interfaces in this layer. In order to make the whole chess-playing process more smooth, an information collection module and feedback module for the robot arm, body and chess-playing tools are also provided.

The command-processing module receives instructions from the manager layer, processes the instructions and then executes accordingly. Such commands as controller parameter configuration, status specification of the robot and chess-playing tools, system initialization, system exit, external output, etc. are executed directly by the command-processing module after the commands are checked and proved to be effective. The commands of motion control are checked, however, for their effectiveness and then the relevant data are put into the task buffer to wait for execution, while the command-processing module continues to process other commands.

After necessary processing, the task-controlling module activates the path-generating task according to task data.

The information collection module of the robot processes the status information of the robot's arm, body and chess-playing tools, and the external analog information from the A/D converter module every 8ms. The information described above is put into the global database for the other modules to call. In order to feedback the signal to the manager layer, the information collection module of the robot transfers collected data at specified intervals to the user control console through shared memory. Therefore, users can obtain the current status of the robot and tools. Again, the status can be saved as files.

The manager layer and task-executing layer are run in WIN32 and RTSS, respectively. It is the RTX's inter-process communication object that transfers data between the two totally different platforms. In order to realize independence and the substitutability property of each module, we designed the same interface for all modules to communicate. The communications details are packaged by the communication interface that allocates each module its exclusive ID number for sending and receiving operation.

4.4. Motion control layer

The motion control layer integrates servo driving, status checking and mechanical execution. The servo system controls the motor and then the mechanical parts according to the data and commands. The status-checking module mainly performs mechanical status checks by means of all kinds of sensors. The RTX technology applied in the robot system improves the whole performance in both real-time property and robustness. Thanks to the compliance of RTX technology with international standards and the abundant API resources offered with the mternational standards, the system is simple in structure and easy to program, debug and maintain. The program can also be easily transplanted. In particular, the RTX's API can be set at different priorities according to different characteristics of data transferring, task execution, etc. In other words, tasks with different real-time demands are assigned with relevant functions, which are favorable to making the system configurable and modular, and the motion control as effective as possible.

5. CONCLUSION

The chess-playing robot is based on a general operating system – Windows NT and RTX from the VenturCom Corporation. High-capability RTX, which transfers various real-time data and commands, can realize real-time position and motion control with high accuracy. The whole system is friendly in graphic user interface, excellent in management and can perform controlling algorithms with high real time property. Modularization in the software design of the control system makes it convenient to modify and append controlling algorithms. Its better expansibility and agility guarantees the real-time property and high-intelligence of the interaction between robot and player. Also, the efficient controller takes full advantage of all kinds of hardware and software resources. The system's clear division, excellent communication, seamless integration and so on are very distinctive and effective; and are the innovation and the characteristics of the whole system. In order to check its practicality, many postgraduates played this new kind of Chinese chess with the robot, and all felt the game was very vivid and the robot very intelligent. In other words, this chess-playing robot and its control system are proved to be

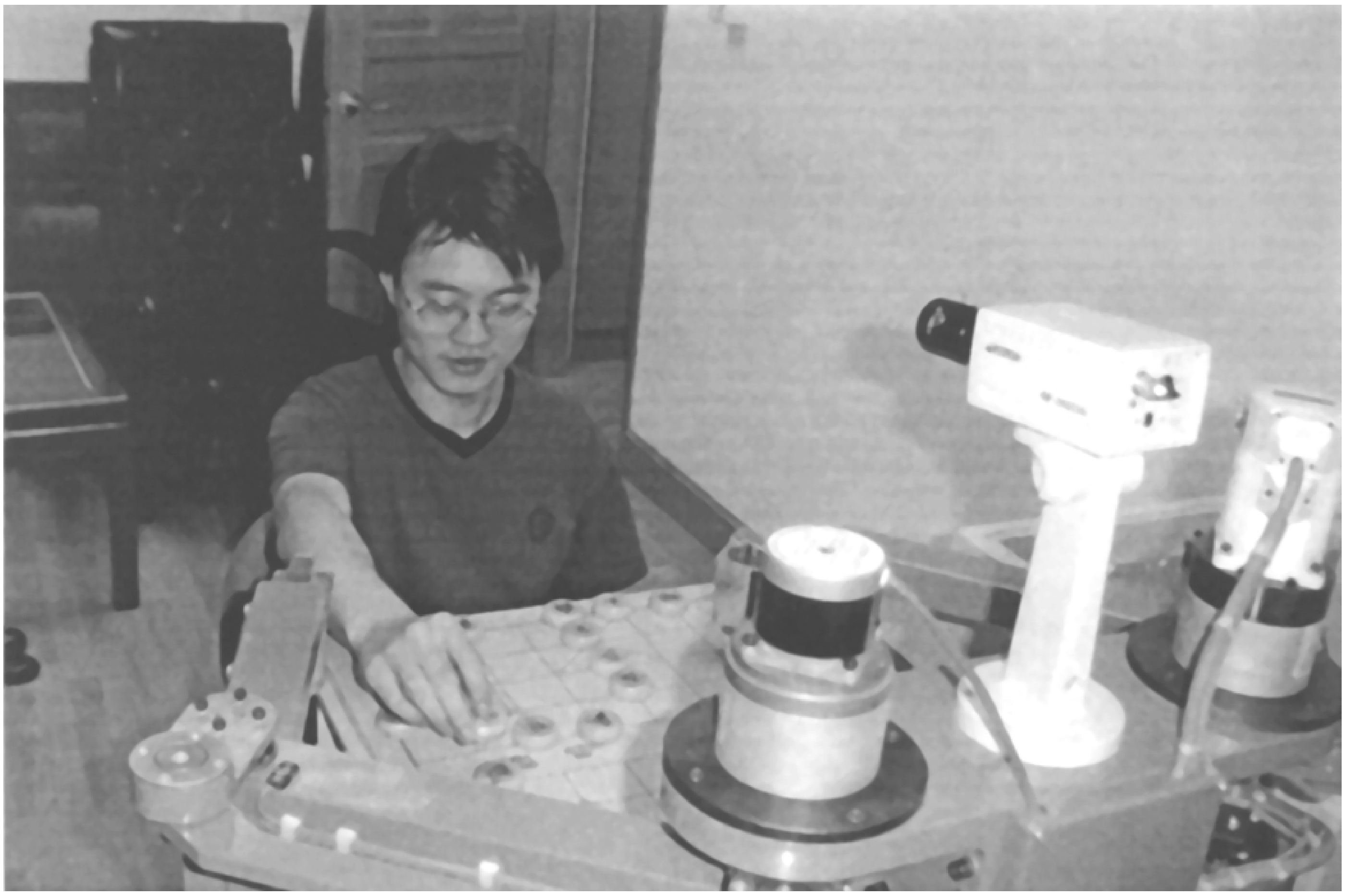

Fig. 5. Illustration of a game between the robot and a player.

reliable, high quality and intelligent. Figure 5 is an illustration of a game between the chess-playing robot and a player.

References

1. Zhang Guangli, Tan Shizhe and Yang Ruqing, "Robot Controller with Open Architecture Based on Windows NT", *Robot* **24**, 443–446 (2002).
2. Fan Yong and Tan Ming, "The Status and Prospect of Robot Controller", *Robot* **21**, 75–80 (1999).
3. VenturCom Inc. RTX 4.3 Users' Guide (VenturCom Inc., Cambridge, MA, 1999).
4. International Standard – Electric Equipment of Industrial Machines – Serial Data Link for Real Time Communication between Controls and Drivers (IEC, 1997).
5. E. Bassi, F. Benzi, L. Lusetti and G. S. Buja, "Communication Protocols for electrical drives", *Industrial Electronics, Control, and Instrumentation, Proceedings of the 1995 IEEE IECON 21st International Conference* in Orlando (6–10 Nov., 1995) **Vol. 1**, pp. 706–711.
6. N. Costescu, D. Dawson and M. Loffler, "Qmotor 2.0 – A real-time PC based control environmemt", *IEEE Control System* **3**, 68–76 (1999).
7. R. Campa and R. Kelly, "An application of real-time control systems to robotics", *Robotica* **19**, Part 3, 323–329 (2001).