Development of a PC-based Open Architecture Software-CNC System

MA Xiong-bo*, HAN Zhen-yu, WANG Yong-zhang, FU Hong-ya
School of Electromechanical Engineering, Harbin Institute of Technology, Harbin 150001, China

Received 7 September 2006; accepted 15 November 2006

Abstract

As a key technology in the field of advanced manufacturing, an open architecture controller is studied. In order to develop an open architecture software-CNC system on personal computer (PC) according to open modular architecture controllers (OMAC). First, the software and hardware platform is chosen and software realization methodology for the CNC system is determined. Second, static modeling methods of an open architecture controller inclusive of object-oriented (OO) programming technology, dynamic link library (DLL) technology and system modules partition are investigated. Third, the dynamical behavioral modeling and the data flow representation of open architecture controller are discussed, which are both described in hierarchy finite state machine (FSM) model. Fourth, a reusable software module model is established to develop software function module library. Finally, a 3-axis milling machine tool test-bed, named for HIT-CNC, is successfully designed by means of the constructed software function module library and the system configuring method. The experimental results show that, besides increasing the degree of reusability and openness, application of above-mentioned methodology leads to significant decrease of development time as well as maintenance cost.

Keywords: open architecture controller; PC; software-CNC; behavioral model; OMAC; machine tool

1 Introduction

Without a unified definition of open CNC system nowadays, openness is generally perceived as modularity, portability, extendibility, interoperability and scalability[1]. In the 1980s, the design philosophy of open architecture controllers began to draw wide attention. It made great progress over the last 20 years as the research on open architecture controllers became an important trend in the field of advanced manufacturing. Provision of a realization basis for advanced manufacturing technology is the purpose of most studies on developing an open architecture CNC controller.

The first open architecture controller was the MOSAIC system developed by New York University in 1998[2]. Since then, ever-more-increasing efforts around the world have been made to introduce open-architecture systems for industrial controls. One of the most important achievements was achieved in 1992 within the frame of European project named OSACA[3]. In 1994, in Japan, a similar project named OSEC under the IROFA Consortium[4] was carried out, and earlier in the USA a number of American researchers acquired outstanding progresses in the realm of OMAC[5].

There were a lot of Chinese experts who engaged in the study of open architecture controllers and thereby devised various controllers on the base of “software IC”[6], software component[7-8] etc., which, however, proved to be defective in incompatibility and lack of portability as well as inter-
changeability. Open architecture controllers are markedly different from today’s vendor-specific, proprietary execution environments and their associated vendor-specific programming environments and interfaces between control elements. Most of today’s proprietary controllers allow neither plug-and-play interoperability, nor sufficient scalability to meet requirements in a broad scope of applications. It is important to note that openness alone does not ensure plug-and-play. Vendor’s idea of openness varies from one to another. As only one step towards plug-and-play, plug-and-play openness, in fact, is dependent on a standard. In order to strike balance between the technology advancements and cost-effectiveness in the open market, open system must be built using “de facto” standards and commercial hardware and software components. Consequently, there has never been a genuine open architecture controller that does not include elements associated with international standards.

By comparison of OSACA, OSEC, and OMAC, Naesa concluded that the three controllers are incompatible in many ways, but OMAC has made efforts to use the results of the other two projects and the OMAC initiative is more promising, though the OMAC API is a little complicated. The international standard of future open architecture controller is most likely to stem from modification of the OMAC API.

In this paper, an open architecture software CNC system named HIT-CNC is developed on PC using OMAC API to obtain the property of “plug-and-play”. Efforts are made to choose the software and hardware platform for the CNC system. Static modeling, dynamic behavioral modeling, data flow representation, reusable software modules and the configuration method are studied. Experiments show that the openness and performance of the system has lived up to expectation.

2 Framework of HIT-CNC

The whole CNC functions are realized by software and the same CPU with the human machine interface (HMI). In opposition to the current situation that investigators are used to relying on difficult-to-reproductive control cards developed by ourselves. In the HIT-CNC system, all motion controls are realized in software. This paper has chosen off-the-shelf hardware to meet the requirements of portability and standardization which enables other groups to copy. The only hardware needed is a single CPU industrial personal computer (IPC) together with a standard communication interface. The basic building block for machine control software is the operating system. The HIT-CNC system adopts Microsoft’s Windows NT with VenturCom’s RTX as its real-time extension to get “hard” real-time capabilities.

The interfaces of control systems can be divided into two groups—external and internal interfaces. External interfaces can be again divided into programming interfaces and communication interfaces. NC and PLC programming interfaces are in agreement with international and national standards, such as ISO6983 (EIA RS274), ISO14649 (STEP-NC), IEC61131-3. Communication interfaces are also standardized to a great extent. Currently, there are several interface standards under development. Field bus systems like SERCOS, Profibus or DeviceNet are used as the interfaces to drivers and I/Os, of which SERCOS is the one where motor drives are connected in a fiber optic ring. This simplifies wiring and abates electrical noise in the system. Because every SERCOS compatible motion controller can be quickly interfaced to the machine tool motion system, and, in some cases, for instance, General Motors has specified the exclusive use of SERCOS as controller platforms for all its new machine tools, SERCOS is selected as the communication interface of HIT-CNC system. Moreover, it is both simple and cost effective to use SERCOS interface in the control with Rexroth’s SoftSERCANS, a master connection for the SERCOS interface which provides a simple and easy way to survey software interface between SERCOS interface ring and control. The software interface makes it easy to implement the SERCOS interface,
for the CNC developer does not have to know which SERCOS interface connection hardware is being used. Therefore, Rexroth’s server systems and motors are selected and the execution elements in HIT-CNC.

Internal interfaces constitute the control system core used for interaction and data exchange between components. To achieve a reconfigurable and adaptable control, based on the platform concept, the internal architecture of the control system aims to hide the hardware-specific details from the software components and to establish a definite yet flexible way of communication between the software components. The application of programming interface (API) fulfills these requirements. As mentioned above, OMAC is selected as the platform of HIT-CNC which involves part of OMAC APIs. The whole function of the control system is subdivided into several encapsulated, modular software components interacting via the defined API.

The general structure of HIT-CNC is shown in Fig.1, where:

1. Software platform: Windows NT4.0+RTX5.1;
2. Hardware platform: industry PC (CPU-PentiumIV: 2.4 GHz, Memory 512 MB);
3. Peripheral equipments: three sets of server driver with SERCOS interface +AC server motor transducer + high-speed electric principal axis and a set of I/O module with SERCOS interface.

The general structure of HIT-CNC is shown in Fig.1, where:

1. Software platform: Windows NT4.0+RTX5.1;
2. Hardware platform: industry PC (CPU-PentiumIV: 2.4 GHz, Memory 512 MB);
3. Peripheral equipments: three sets of server driver with SERCOS interface +AC server motor transducer + high-speed electric principal axis and a set of I/O module with SERCOS interface.

3 Static Modeling of HIT-CNC

3.1 OO technology

OMAC API uses an OO approach to specify the modules’ API with class definitions. A class is defined as an abstract description of the data and behavior of a collection of similar objects. Classes aggregate data and method. Class definitions offer encapsulation hiding details of a class. For example, SERCOS-Driven Axis describes an instance of an Axis class in the running machine controller. A 3-axis milling machine would have three instantiations of that class—the three objects implementing that class. An object-oriented program is considered to be a collection of object interacting through a set of published APIs. A byproduct of the object-oriented approach is data abstraction, which is an effective technique for extending a type to meet program needs.

Inheritance is useful for developing data abstraction. OO classes can inherit the data and methods of another class through class derivation. The original class is known as the base or super-type class, while the class derivation as a derived or subtype class. The derived class can add to or customize the features of the class to produce either a specification or an augmentation of the base class type, or simply to reuse the implementation of the base class. To achieve an object-oriented framework strategy, all OMAC API class signatures (methods) are considered to be “virtual functions”, which allow the derived classes to redefine an inherited base class method.

The foundation classes are derived from decomposing a generic controller into classes, which define the controller class hierarchy as shown in Fig.2. Foundation classes are then grouped into modules that become plug-and-play components.
3.2 DLL technology

The OO technology can gain reusability in source code level; another kind of software reuse method is based on binary code reuse which includes the reuse of executable program and function library. DLL and Microsoft’s component object model (COM) belongs to the latter and can easily satisfy different application requirements of open modular software-CNC controller such as turning, milling and drilling. Module is the base unit composed of an open CNC system. The modules are of two types: non-real-time and real-time, which are respectively realized in COM component and RTDLL (real-time DLL) according to the tasks’ real-time capability.

The Microsoft COM is a frame work for creating and using components. As a Microsoft architecture for local interaction of components, COM serves to facilitate component technology including location transparency, security, registry, naming, and type information. COM is also a kind of DLL, which supports location transparency by allowing components to be deployed as in-process DLL, or as local servers. The in-process COM components impose practically a zero sacrifice on performance, while local servers use inter-process communication (IPC). The non-real-time tasks include inputting and editing part program, displaying system and manufacturing information, translating part program, and tool offsetting etc. Windows’ timers can meet the time requirements of these tasks. Thereafter the tasks are wrapped into modules realized in COM component technology.

The real-time tasks of CNC refer to the disposal of acceleration/deceleration, interpolation, position control, discrete logic control etc. These hard real-time tasks must run in the RTX environment to fill their time requirements. RTX supports OO programming technology and provides a wizard kit in Visual C++ environment. The hard real-time tasks are developed in RTDLL provided by RTX.

Similar to Windows’ DLL in conception, RTDLL can be dynamically loaded in RTX environment. By extending Windows’ operating systems, RTX enables application components or modules that require deterministic and high-speed response times, and other non-real-time application components to work together in a common Windows’ system.

RTDLL’s real-time capability is guaranteed by the real-time APIs provided by RTX. RTX adds a real-time subsystem, known as RTSS, to Windows NT and Windows 2000, XP. In conception, RTSS is like other Windows NT subsystems (such as Win32, POSIX, WOW, and DOS) in that it supports its own execution environment and APIs. However, the main difference lies in that RTSS, obviating the Windows scheduler, performs its own real-time thread scheduling. Furthermore, in a uni-processor environment, all RTSS thread scheduling occurs ahead of all Windows scheduling including Windows-managed interrupts and deferred procedure calls (DPCs). RTX supports various runtime libraries, and provides a ‘C’ runtime library based on MS Visual C++. RTSS processes can be statically linked so as to include these libraries unless they do attempt to link to unsupported Win32 functions. RTSS processes can also be linked into specialized versions of dynamic link libraries (DLLs), which can be used to modularize real-time application code or provide run-time customization of real-time software environments.

3.3 OMAC and its API in HIT-CNC

The OMAC users group is of an industry forum aimed at advancing the controller technology\textsuperscript{[9]}. An effort has been made within OMAC to define API specification for the purpose of submitting it to an established standards body one day.

The OMAC API adopted a component-based approach to achieve plug-and-play modularization by making use of interface classes to specify the API\textsuperscript{[9]}. For distributed communication, component-based technology uses proxy agents to tackle method invocations that cross process boundaries. OMAC API contains different “sizes” and “types” of reusable plug-and-play components inclusive of
component, module, and task—each containing a unique finite state machine (FSM) model to perform component collaboration in a definite manner. The term “component” applies to reusable pieces of software acting as a building block in use, while the term “module” refers to a container of components. Tasks amount to those that are used to an encapsulate programmable functional behavior comprising a series of steps from start to finish supporting starting, stopping, restarting, halting, and resumption, and may run multiple times while the controller works. Tasks can also be used to build up controller programs consisting of a series of tasks with ability to restart and navigate, or as standalone resident tasks to cope with specialized controller requirements.

To integrate components, a framework is necessary to formalize the collaborations and other life cycle aspects in which components operate. The OMAC API uses COM as the initial framework to develop components so that control vendors could concentrate on application or specific improvements to define their strategic market-share as opposed to spending valuable programming resources reinventing and maintaining software “plumbing”[1]. The lack of hard, real-time preemptive scheduling once posed the primary problem with COM framework, especially for the Windows NT operating system, but it is resolved in HIT-CNC by third party extensions of Windows NT as RTX.

International interfaces served interaction and data-exchange between components that form the control system core. An important criterion in this area is the support of real-time mechanisms.

Fig.3 illustrates modules in HIT-CNC according to OMAC functionality. The HMI module responsible for human interaction has a controller which presents data, treats commands, and monitors events and in the OMAC API “mirrors” the actual controller with references to all the major modules and components via proxy objects. The Task Generator module serves to translate the application of part programs written in the RS274 or ISO 14649 into control plans which include motion segment instructions. The Task Coordinator module is responsible for sequencing operations and coordinating the various modules in the system based on programmable tasks. The Task Coordinator can be regarded as the highest level finite state machine in the controller. The Axis Group module is in charge of interpolating and coordinating the motions of individual axes, transforming an incoming motion segment specification into a sequence of equi-time-spaced set-points for the coordinated axes. The Axis module is held responsible for servo control of axis motion, transformation of incoming motion set-points into set-points for the corresponding actuators. The Control Law component acts as servo control loop calculations to research specified set-points. The Soft PLC module is employed to implement discrete control logic or rules characterized by a Boolean function from input and internal state variables to output and internal state variables, and to realize the reading of input devices and writing of output devices through SERCOS IO module interface.

4 Dynamically Behavioral Modeling and Data Flow Representation

4.1 Behavior model

For the OMAC API, behavior of the controller is embodied in FSM. An FSM step represents a situation in which the behavior in connection with inputs and outputs follows a set of rules defined by the associated actions of the step. A step is either active or inactive. Action is a step a user takes to complete a task that may invoke one or more func-
tions. A transition represents the condition, through which the control passes from one or more steps preceding the transition to one or more following it.

In an OMAC API module, there may exist nesting of FSMs. OMAC API does not dictate the number of levels of FSM. In general, there is an outer administrative FSM to handle activities inclusive of initializing, startup, shutdown, and, if necessary, power enabling. The administrative FSM must follow established safety standards. When the administrative FSM is in the READY state, FSM is possible to descend into a lower level. The theory of nesting FSMs used in HIT-CNC is described in detail in Ref.[12].

4.2 Data flow control and execution FSM

Besides the transmission of events, there is the transmission of data between modules.

1. Abstraction of data flow and customization of task unit

The control information, in which are encapsulated a series of state changes that might be undergone and operations that might be expected to fulfill, is extracted from part program in Task Generator module, which is also described in FSM and thus possesses intelligent characters. The FSM object which contains control information is named task unit (TU).

OO programming technology is adopted to abstract TU. Fig.4 illustrates the hierarchy of TU specialization in HIT-CNC, among which TU is the base class. TU specification is the mechanism to add extensions. For example, the NURBS motion segment is derived from Motion Segment TU. Capability is a derived class of TU used within a Task Coordinator corresponding to different machine modes (manual, auto). When the capability FSM is in the READY state, the capability can descend to a lower FSM or TU. For example, once in the auto capability FSM, a lower level FSM for the “cycle” TU can be used to sequence through a series of TUs.

Execution Step is the execution FSM in Task Coordinator, monitoring the execution of Motion Segment and Discrete Logic Unit. Motion Segment corresponds to the FSM input for an Axis Group module. In addition to the FSM directive and parameter methods, a Motion Segment includes information such as rate, geometry and a reference to a velocity profile generator necessary for trajectory planning. Acceleration/deceleration and interpolation are implemented by Motion Segment. Discrete Logic Unit corresponds to the FSM input for a Soft PLC module. Discrete Logic Unit coordinates and controls an aggregation of IO points. In addition to the FSM directive and parameter methods, a Discrete Logic Unit contains the information necessary for defining either asynchronous logic—the event or condition trigger, or synchronous logic—the scan rate and FSM.

2. Nesting of Task Unit and collaboration of modules

A TU may contain other TUs, in other words, TU can be nested. When activated, a TU can send embedded TUs to lower level servers. Thus, TU is “intelligent” to understand how to coordinate and to sequence the lower level logic and motion modules. Fig.5 illustrates the relationship between a TU’s states and its travel through HIT-CNC.

TU is expressed as data transmission among modules in steps as shown in Fig.6. Step 1: part program in Task Generator module is translated into a series of nesting Execution Step TUs which include Motion TUs or Logic Control Units to be executed. Step 2: Task Coordinator accomplishes Execution Step TU via getTasks( ). It is worth noting that Task Coordinator is a RTDLL running in the environment of RTSS while Task Generator is realized by making use of COM technology running in the environment of Win32 without a “hard” real-time capability. They are synchronized via the inter-
process communication objects such as mutex, semaphore and transmit Execution Step TU via RTSS shared memory objects which allow sharing blocks of data among multiple processes including RTSS processes and Win32 processes. Step 3: Task Coordinator activates the FSM of Execution Step Unit via its executeUnit( ). This step may run repeatedly as it must synchronize with its lower tasks, for example, waiting for the end of current nest Motion TU. Step 4: the activating Execution Step TU transmits its nested Motion TU to the motion queue in the Axis Group module via setNextMotionSegment( ). And then the Motion TU waits to be activated. Step 5: once the requirements of activation are met, the Axis Group module calls the executeUnit( ) method of Motion TU periodically and the FSM of Motion TU becomes the execution FSM of the module. And the FSM of outer Execution Step TU is still running in the Task Generator simultaneously until the Motion TU turns into “Done” state.

The state transitions in the process of the transmission and execution of the TUs in the control system mentioned above are described in Fig.6, with their management objects changing at the same time. The nesting TU is created by the translation of the Task Generator and its FSM is activated by the Task Coordinator by calling the executeUnit( ) method of outer Execution Step TU. Then the Task Coordinator refreshes the Execution Step FSM which results in the motion segments being sent to the motion queue of the Axis Group and the Execution Step’s state being transited into “running” while the state of the motion TU nested in the Execution Step TU keeps stable. Thereafter, the executeUnit( ) method of the Motion TU is called by the Axis Group and activated in the Axis Group to be the execution FSM. And then it is refreshed periodically to deal with the “init”, “run” events in turn to trigger the interpolation arithmetic of Motion TU. The Axis Group judges the end of Motion TU by the calling of Done( ) method. In this process, the Execution Step task which is controlled by the upper Task Coordinator keeps the “running” state, until it is notified of that its inner Motion TU has finished its tasks. If the Motion TU ends, the Axis Group module deletes it from its dominion. Then the Execution Step watches the next motion segment or it takes control of the next Execution Step.

5 Model of Reusable Software Unit

Based on the static model and dynamic behavioral model constructed above, the reusable software module structure, shown in Fig.7, describes modules’ inner function in hierarchical FSM and provides outer service by three types of interface (infra-structure, connection and function interface).
Fig. 7 Reusable modules structure.

The infrastructure interface performs operations related to the registration, setup, initialization, safety and error disposing. The connection interface, which determines a framework for connection, defines the relationship between itself and other modules. The function interface encapsulates specified CNC functions the modules contained, which include various methods to operate the action, state and parameter of the module. The FSM is composed of the administration FSM and the module’s own FSM. Clients can transmit FSM object to the module or trigger the generation of its inner events by calling the service provided by the module through its interfaces and then the module schedules and coordinates the running of the related dominated object to fulfill clients’ command requests according to the FSM, at the same time the module can issue its command request to other modules.

To improve the efficiency of the developed software modules and facilitate the integration of open architecture CNC, the interfaces of modules must have a general format compatible with each other. A basic software module library composed of HMI, Axis Group, Axis, Task Generator, Task Coordinator, Soft PLC modules and etc. is developed following this structure.

6 Configuration of HIT-CNC

One purpose of establishing an open architecture controller is to quicken responses to the changing market requirements from end users. Adding or changing functions of an open software CNC system is an important means to supply end users with the possibilities to customize a system, which, in fact, matters a great deal to an open software CNC system[13].

Configuration is defined as a system module customization to realize a specific demand on a general solution scheme. The procedure of using a series of function modules to construct a CNC system for a specific machine tool is as follows: selecting modules from the software module library in accordance with the type of machine tool, and then customizing the modules’ function followed by constructing the communication connection of the modules[14].

End users’ configuration request is kept in a document (*.SCF), which, actually, is the configuration set of modules to construct a new CNC system. The Axis Group’s configuration document of HIT-CNC is as follows.

```
[AxisGroup_0]
ModuleType = RTDLL
 Lib_Name = axisGroupModule.rtdll
InterfaceType = CAxisGroup
Number_Axes = 3
Cycle_Time = 0.002
ACC_DEC = Trapezodial
X_Axis = Axis_0
Y_Axis = Axis_1
Z_Axis = Axis_2
```

The Axis Group’s configuration document shows that the module is realized in RTDLL, and the module library is named axisGroupModule.rtdll, which serves other modules through the CAxisGroup interface. Connected to three situations of axis module, the Axis Group is responsible for coordinating the motion of the three axes. The interpolation period of the system amounts to 2 ms, and the acceleration/deceleration mode is trapezoidal. In the Axis Group, the connected axis module is named “X_Axis” locally and “Axis_1” globally.

7 Accessing of Functionality and Openness of HIT-CNC

7.1 Set-up of HIT-CNC

On the base of the software module library,
HIT-CNC, is set up successfully an open architecture software CNC system, which has friendly HMI, tool offset function, supports line, arc, and NURBS interpolation method. Integration is defined as the capability to allow the connection and cooperation of two or more modules within a system. The integration procedure of HIT-CNC is shown in Fig. 8.

HIT-CNC is a 3-axis milling machine tool and the configuration document is edited as 3_Axis-Mill.SCF.

After start-up of the CNC system program, the SCF document is parsed automatically, and the specified function modules are dynamically loaded according to the definition of the configuration document. Thereafter, the parameters of the modules are configured, connected, and started to set up a new CNC system. In the end, the system sets working and fulfilling related CNC functions following the end users’ directions.

The prototype 3-axis milling machine equipped with HIT-CNC is shown in Fig. 9.

### 7.2 Manufacturing experiment in HIT-CNC

Fig. 10 is a mouse cover machined in HIT-CNC using line and arc interpolation function.

Fig. 11 is a rough-machined piece of an electric drill die in HIT-CNC using 3-axis NURBS interpolation function.

Fig. 12 is the finish-machined electric drill die.

### 7.3 Openness of the system

Configurable and reusable software modules are selected as the basic units to construct an open architecture software CNC system, HIT-CNC, which present satisfactory openness inclusive of modularity, reusability, reconfigurability, portability, extendibility and scalability as well as plug-and-play property.
(1) Performance parameters like the interpolation period, servo period, acceleration/deceleration mode can be customized by end users.

(2) The framework parameters like the controlled axis number of the system can be configured.

(3) The extendibility of the system is maintained so that the system’s function can be remodeled by adding, truncating, or replacing modules. For example, a new turning machine tool CNC system can be easily produced by substituting a turning Task Generator Module for the milling Task Generator Module. Also a new interpolation method such as NURBS interpolation can be integrated into the system.

8 Conclusion

The open architecture controller is emblematic of one great advancement in the field of advanced manufacturing technology. The research of open architecture controllers has its purpose of providing a realization basis for advancing it. In this paper, after close scrutiny of the ways to realize the basic functions, a whole new structure of the controller is designed and a prototype is worked out on PC on the ground of OMAC. The experimental results corroborate the gain of a satisfactory openness characterized by modularity, reusability, reconfigurability, portability, extendibility and scalability.

References


Biographies:

MA Xiong-bo  Born in 1976, he received B.S. from Harbin Institute of Technology (HIT) in 1999, and became a Ph. D. candidate of HIT in 2004. And now, his research interests cover open architecture CNC system and CIMS. E-mail: rogers2008@126.com

WANG Yong-zhang  Born in 1941, as a professor and doctoral tutor in HIT, till now, he has completed 22 scientific research projects and had 5 monographs and more than 70 academic papers to his credit. His research interests cover NC technology, application of computer in machining and design/ machining/ controlling of modern manufacturing systems. E-mail: nc_mach@hit.edu.cn.