OPEN ARCHITECTURES FOR MACHINE CONTROL

by

Frederick M. Proctor, Brad Damazo, Charles Yang, and Simon Frechette

National Institute of Standards and Technology

Gaithersburg, Maryland 20899

ABSTRACT

A major impediment to improving the performance and functionality of machine tools is the limited access that users or third parties have to the internals of the machine controller. This limit has suppressed the emergence of a community of third-party vendors who could provide a wide range of applications at competitive prices. The result is that users are often faced with all-or-nothing compromises when choosing controllers, and are restricted to the original controls vendors for everything from spare parts to software. Machine tool users would benefit from an open architecture that can serve as both a target for innovative third party product development, and as a specification which produces multiple competitive sources for interoperable products.

The National Institute of Standards and Technology has initiated a project which will demonstrate the feasibility of open architectures for machine control. This project, the Enhanced Machine Controller (EMC), has selected several target applications which improve the accuracy and ease of use of machine tool controllers: selectable look-and-feel, tool management, alternate part programming languages, spline-based motion, in-machine inspection, and thermal-geometric error compensation.

1. INTRODUCTION

The National Institute of Standards and Technology (NIST) has been working with a variety of industry representatives and organizations to investigate the application of open architecture concepts to machine tool controllers. The objective of the NIST effort, the Enhanced Machine Controller program, is to develop methods which will reduce the cost of controller development. The cost reductions are intended to impact members of several categories: machine tool builders, controller retrofitters, system integrators, and end users. As a means to this end, the EMC program has defined an architecture which supports the integration of commercial off-the-shelf components [1]. The resulting benefits are numerous:

- machine tool builders and controller retrofitters would enjoy competitive sources for controller components, such as computing hardware, motion control hardware, and input/output systems, which would be compatible with one another;
- system integrators would be able to seamlessly connect machine tool controllers into an effective factory enterprise;
- maintenance personnel could select from a variety of spare parts sources to meet their costs and schedules;
- entrepreneurs would have a target for commercializing new technology.

The EMC program is targeting several applications that will illustrate the benefits of the architecture, and demonstrate what is required to support their cost-effective integration. To this end, NIST has established a testbed that will focus on integrating existing commercial components and

enhancing them with additional capabilities. Ultimately, the same rapid revolution that typified office computing in the past decade would take place on the shop floor, and capabilities that are not yet concepts would become commodities.

2. OBJECTIVES

The objective of the EMC program is to develop methods to reduce the cost of controller development. These methods must support those who want to piece together their own systems component by component, who want to modify or improve their controller, who wish to apply existing modifications from one open controller to another open controller, or who intend to start small and upgrade as they grow. These requirements mean that the controller architecture must be modular, extensible, portable, and scalable.

2.1 Modularity

Modularity refers to the ability of machine tool builders, retrofitters, and maintenance personnel to purchase and replace components without unduly affecting the rest of the controller. For the EMC, this means the ability to replace motion control boards, motherboards, disks or RAM with equivalent products from another source. The key to being able to do this is the definition of the modules which make up a controller, their functionality and interfaces, and the basic controller infrastructure (buses, CPU chipsets, operating systems) which support them.

2.2 Extensibility

Extensibility gives users and third parties the means to incrementally add functionality to a module without replacing it completely. For example, an extensible part program interpreter would allow developers to add new statements to the language. If the part programming language were EIA-274-D [2], users could add a new M code applicable to custom tooling. An extensible input/output system would allow the easy addition of the actuators or sensors associated with the custom tooling.

2.3 Portability

Portability makes it easy to transfer applications or enhancements which have been developed on one controller to a controller based on another platform. Portability normally refers to software source code, which should be easily compilable to run on a variety of CPUs and operating systems. Portability is of primary importance to developers of software that is intended to run in an open controller. If controllers conformed to common and public specifications, then the effort required by entrepreneurs to support different manufacturers would be drastically reduced.

2.4 Scalability

Scalability gives machine tool builders and retrofitters the ability to build controllers around their customer's needs and budget. This gives customers the freedom to "scale up" to a higher-performance implementation as their needs and budgets increase. For example, if a particular shop floor application were undemanding, a low-cost solution would be available. If, at a later date, the application's performance requirements increased, portions of the controller could be upgraded to meet these new demands.

3. EMC ARCHITECTURE

The EMC modularity objective implies that the architecture must specify the components which make up a controller, and their interfaces and behavior. The extensibility objective implies that each components' functions and data must be made available. Portability and scalability imply that controllers be comprised of commercial off-the-shelf components, and that controller services such as the computing hardware, mass storage devices, and operating systems be non-proprietary.

A controller based on a candidate architecture has been implemented on a Laboratory Development Controller, described in Section 4.2. This architecture is derived from the NIST Real-time Control System reference model architecture (RCS) [3] and NASREM, the NASA/NBS reference model architecture [4]. RCS builds upon experience acquired in the Automated Manufacturing Research Facility [5] during the 1980's for constructing hierarchies of controllers for machining workcells which include robot, machine tool, and coordinate measuring machines as components.

The architecture defines the following components: Workstation Planning, Workstation Management, Plan Interpretation, Discrete Input/Output, Trajectory Generation, and Servo Control, as shown in Figure 1. These components communicate by message passing. A distributed world model is comprised of data made public by each component, and made available through messages which return the values of data. In the figure, the blocks to the right indicate independent applications which are free to communicate with the specified components, and provide some additional functionality or enhancement. An example of such an application is the Operator Interface, which reads values from each component for display, and sends messages to each component requesting action. Each of the components is described in the following sections.

3.1 Workstation Planning

Workstation planning is responsible for taking a part design and generating the sequence of operations which will transform stock into a finished part. In a traditional system, planning may involve the generation of numerical control (NC) programs by a PC-based CAD/CAM system. In more complicated scenarios, the workstation planner may rely on scheduling and process planning software, solid modelers, and expert systems to automatically produce a complex part requiring machining on several tools and robot part handling and assembly. If the workstation planning component can accept feedback from the controller, more intelligent machining may be accomplished. For example, if the programmer specifies a 2-inch end mill during the NC code generation phase, and only a $1^3/_4$-inch end mill is available at run time, the part programmer is required to repeat the programming phase. If the workstation planner can accept feedback, a tool mismatch could signal the workstation planner to automatically regenerate the NC code based on the existing tools.

3.2 Workstation Management

Workstation management is responsible for maintaining the modes of the controller. In a workcell with a single machine, such as a three-axis mill, the job of the workstation management component consists mainly of arbitrating between automatic and manual mode, and insuring that actions that share resources do not occur simultaneously. For example, a typical workstation management component would insure that the operator could not jog the axes during machining, or that tool changes can only be requested when the spindle has been retracted. If the workcell consists of several machines operating in tandem, such as a machine tool and pallet loader, then more complicated management is required.

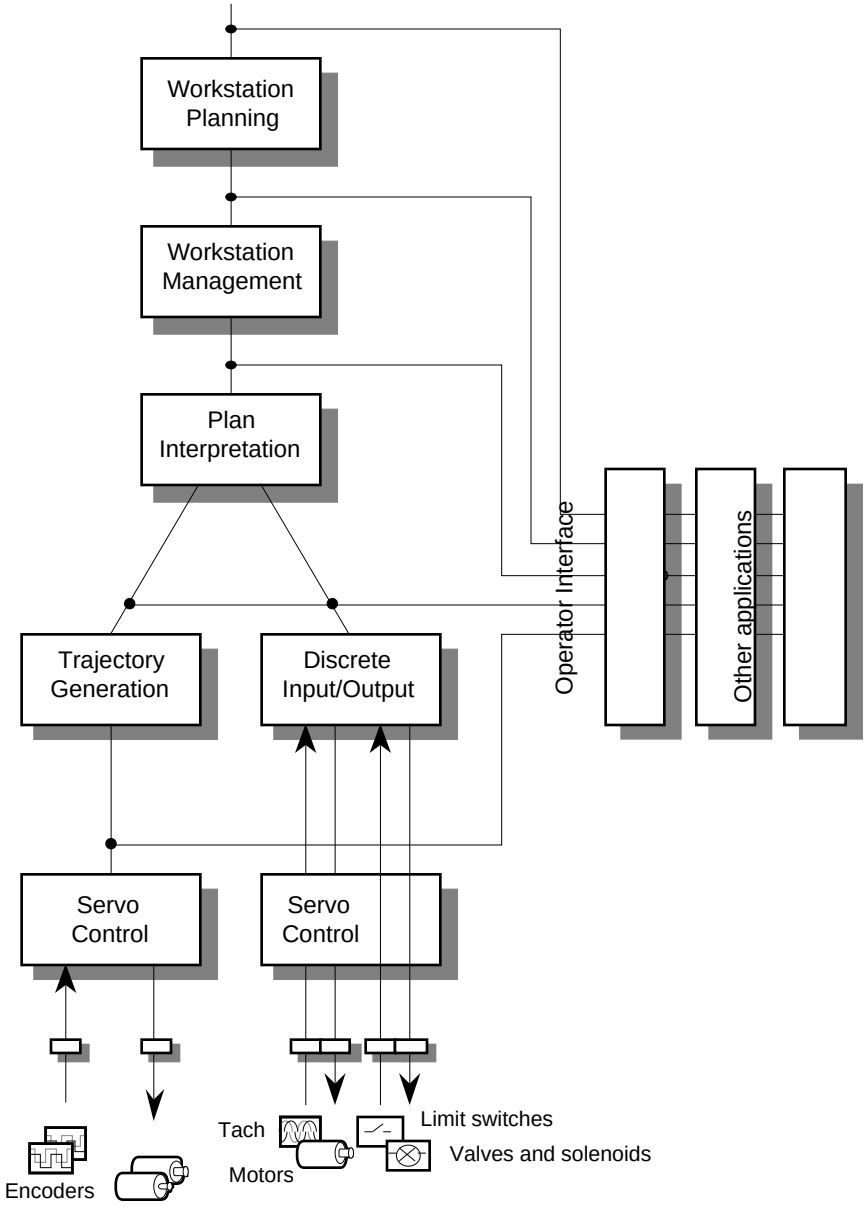

Figure 1. The Enhanced Machine Controller architecture.

3.3 Plan Interpretation

Plan interpretation reads and executes the sequence of machining instructions. Traditionally, these instructions would be NC code files. The actual functions are carried out by the plan interpreter's subordinates. The interpreter must insure that instructions have been executed to completion, if that is required by the semantics of the programming language. In the EMC architecture, the function of plan interpretation has been modularized so that a single controller can at different times execute part programs written in different languages, such as EIA-274-D or Binary Cutter Location (BCL) [6].

3.4 Discrete Input/Output

The discrete input/output component is responsible for executing programs that read input or compute output for the sensors and actuators that are not synchronized with coordinated axis motion, for example, limit switches and valves for tool changers. Traditionally, this component has been termed the programmable logic controller, or PLC, and its programs have been written in ladder logic. Modularizing this functionality allows system integrators to include I/O systems from a variety of vendors, or to program them in languages that suit their experience or the experience of the field service or shop floor personnel.

3.5 Trajectory Generation

The job of the trajectory generation component is to plan and execute coordinated axis motion, such as linearly interpolated moves or circular arcs, specified by the part program. Although the bulk of NC programs include only straight lines and circular arcs, industries such as aerospace design more complex shapes that do not lend themselves to these simple motion primitives. Furthermore, CAD/CAM systems have increasingly been incorporating smoothing techniques which generate mathematically more complex curves. Approximating these curves with straight line segments leads to a trade-off between accuracy and size of the part program: the greater the desired accuracy, the more segments required for the approximation and the longer the part program. Since there appears to be a trend to design more complex parts, there is a strong justification for modularizing the trajectory function so that it may be enhanced with more complex motion primitives should the application demand it.

3.6 Servo Control

The trajectory generation component computes a sequence of positions (plus velocities or other quantities, depending on the sophistication of the controller) to the servo control component, which converts them to axis positions and issues control signals to the motors. Typically, machine tool servo controllers use a proportional control law. In more difficult control applications entirely different control laws may be desired, such as pole placement or gain scheduling. Alternatively, if the process is undemanding, open-loop motor control may be preferable, for example using stepper motors. In any case, modularizing the servo control function allows control builders to choose which motion equipment is the most suitable for the job, given constraints such as cost, power consumption, and packaging. Providing extensibility in the servo control component would allow service personnel to modify the control gains to tune the performance, or would enable sophisticated users to install different control algorithms altogether.

3.7 Independent Applications

Just as the desktop computing industry has enjoyed a proliferation of hardware and software products which have originated from a variety of entrepreneurial sources, it is expected that manufacturers will benefit from enhancements developed by innovative third parties for open architecture controllers. Instead of trying to exhaust the range of possible enhancements, and set aside places for them in the architecture, the approach was to limit the core components to those that are necessary in most machine tool controllers today. Excluded applications should not suffer from this approach, since they are free to communicate with the core components described above in exactly the same way as the core components communicate between themselves.

The EMC program has investigated several enhancements, primarily to determine what is required to make the job of integrating independent applications easier and less costly. In the next section, the NIST controller implementations are detailed, to give the enhancement descriptions some context.

4. SYSTEM DESCRIPTION

The controller implementation efforts at NIST are divided between two platforms, the Shop Floor Controller and the Laboratory Development Controller. These platforms serve as testbeds to demonstrate the benefits that can be derived from enhancements, and support investigations into reducing the cost of integrating commercial components.

4.1 Shop Floor Controller

Many enhancements rely on sensor input or process measurements which take place on the shop floor during machining, and must be developed *in situ*. In these cases, the efforts are focused primarily on the technology and algorithms, and take place on the Shop Floor Controller [7]. The Shop Floor Controller is based on the PMAC motion controller from Delta Tau Data Systems[1], in an environmentally-enclosed 486 PC. The 486 is running the DOS operating system. This controller is connected to a Monarch VMC-75 3-$^1/_2$ axis vertical machining center, which has a positioning W axis and a tool changer. A VGA monitor, keyboard, and trackball were mounted into an enclosure for the operator console, replacing the GE Mark Century 2000 controller console. In addition, the control console includes jog buttons, a handwheel, and mode select switches. All the Monarch control input and output lines run into a terminal block inside a cabinet at the back of the machine, and are interfaced to the PMAC motion controller using a variety of interconnect modules provided by Delta Tau. The Shop Floor Controller is depicted in Figure 2.

In the Shop Floor Controller, the host computer is running Delta Tau's PMAC-NC program, which provides operator interface functions such as graphics display and control panel input, manages data such as tool offsets and parameter tables which affect axis motion, and manages the rotary buffer used to download part programs to the PMAC board. NIST developed graphics "CNC front ends" for this controller which can emulate either a Fanuc 10, GE 2000, or Allen-Bradley 8200 interfaces. EIA–274–D part programs are first parsed by the PMAC-NC program, and translated into the PMAC native language. The PMAC board communicates to the host PC using a dual-ported RAM mechanism. The PMAC board performs the coordinate system trajectory planning, the axis trajectory interpolation, the servo computations, all of the PLC tasks (ZW axis selection, control panel I/O, and tool changer), and axis limit checking.

4.2 Laboratory Development Controller

Once the technology underlying an enhancement has been proven, the application programs and any support hardware necessary are transferred to the Laboratory Development Controller, which focuses on software engineering. Here, any platform-specific references are noted and localized so that the application can be made portable.

---

1. Equipment listings are provided throughout this document for the sole purpose of clarifying the discussion, and in no way imply recommendations by NIST.

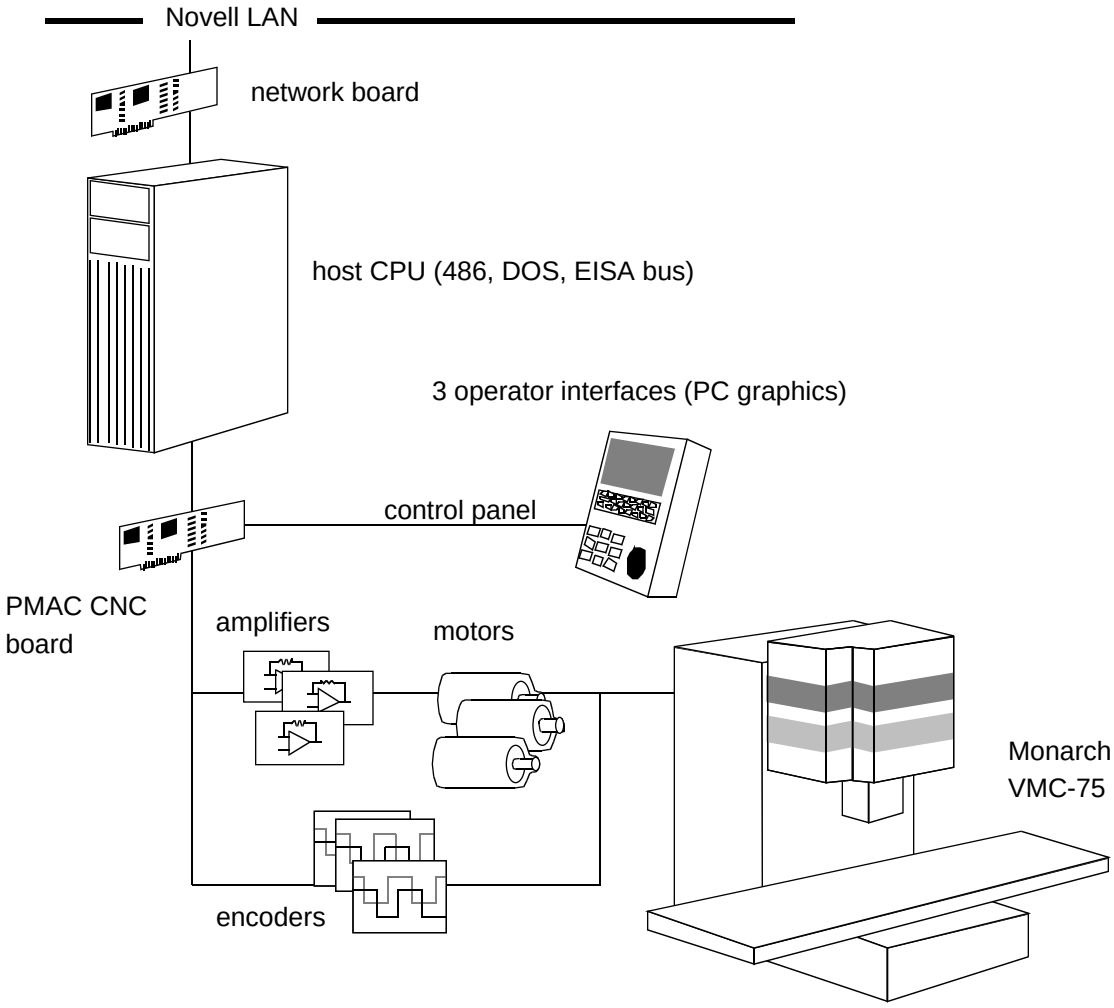

Figure 2. Shop Floor Controller.

The Laboratory Development Controller is hosted by a 486 PC, like the Shop Floor Controller. However, this system differs in several ways. Most importantly, it is not connected to a production machine tool, but to a desktop three-axis mill which contains DC servomotors, incremental encoders, and a 3000 rpm spindle suitable for cutting machining wax or soft aluminum. This allows development work to proceed without anxiety about injury or damaging expensive production machinery. The Laboratory Development Controller is running a real-time Unix operating system which is POSIX compliant, and uses X Windows and Motif for graphics services. The Laboratory Development Controller is depicted in Figure 3.

Most of the functions performed by the PMAC board in the Shop Floor Controller are performed in software on the PC host, namely PLC programs, part program interpretation, and trajectory generation. PLC programs are written in the C language, and read and write to a commercial off-the-shelf digital input/output board through the PC bus. Part programs are first translated from EIA-274-D into a C language program which is then compiled, and are executed in software by the trajectory generator component. Two trajectory generators are supported, dem-

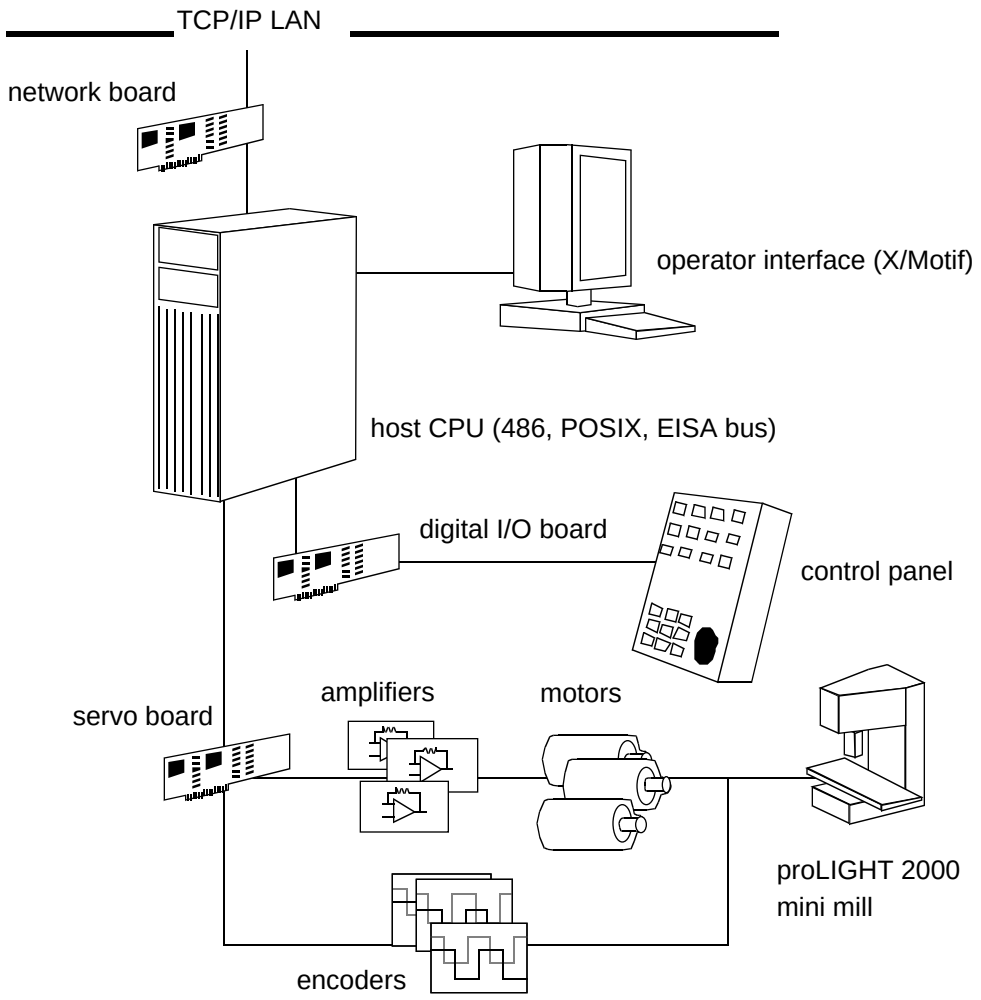

Figure 3. Laboratory Development Controller.

onstrating modularity: Level 2 from the Indiana Business, Modernization, and Technology Corporation, and TMOS from Trellis Software and Controls. The trajectory generator communicates via message passing through shared memory to the servo control component, which is comprised of a software interface and hardware support. The software portion has been written to support modularity, and supports the PMAC board as well as the Matrix4 board from DSP Control Group.

5. ENHANCEMENTS

To validate that the architecture is all it purports to be, NIST placed itself in the role of system integrator to incorporate some enhancements into each controller. These enhancements included selectable look-and-feel, tool management, alternate part programming languages, spline-based motion, in-machine inspection, and thermal-geometric error compensation.

5.1 Selectable Look-and-Feel

Consider the benefits of emulating the look and feel of a suite of existing controller displays on a single controller. A machinist familiar with, say, the Fanuc interface would feel at home one minute, and the machinist on the next shift could select the Allen-Bradley look and continue working on the same machine. This capability can be a significant advantage: in some cases, controllers which have exceeded their lifetimes have been replaced with identical but technologically obsolete controllers to eliminate the need for expensive operator training. If a newer and technologically superior controller were open, operator interfaces emulating the older controller could have been developed, bringing the benefits of new technology while eliminating training costs.

In the Shop Floor Controller, NIST programmers have emulated the look-and-feel of three common commercial operator interfaces: the Fanuc 10, Allen-Bradley 8200, and General Electric Mark Century 2000. The programs for each may be run independently of the controller; in fact, the operator may terminate one and run another during machining. The programs are written in the C programming language. The Fanuc 10 interface has also been ported to the Laboratory Development Controller, rewritten to use the X Windows and Motif standards.

There are several key requirements of an open architecture to support the development of independent graphical user interfaces. First, the controller must make available to independent parties all the mode and state information that machinists expect to view at the display during machining. Based on an analysis of the existing operator interfaces, this information includes:

- current machine positions: relative, absolute, distance to go
- feedrate
- spindle speed
- actual feedrate
- actual spindle rate
- increment value
- program name and text
- current line executed
- tool offset
- work zero offset
- operator messages

Additional diagnostic and maintenance information must also be made available. This information includes

- controller gains
- velocity profiles
- acceleration times
- state of discrete input and output points

Another area of operator interface that has been explored is the operator panel, which typically provides jog buttons, mode switches, and other “hands-on” items that machinists use to direct the operation of the machine. NIST engineers designed a custom, non-proprietary operator panel which provides all the common functions available on the previous controller. Unlike the graphic

display, however, the panel is not subject to “on-the-fly” selectability by the operator, since it is a hardware device. However, discussions with machinists have indicated that the most difficult information to assimilate when learning a new controller is not the physical, static layout of the operator panel, but the layout of the many individual screens which make up the graphic display. This indicates that for the machinist, a tremendous improvement can be realized solely in software.

5.2 Tool Management

Effective tool management is a vital part of a production control system [8, 9, 10]. The overall goal of any tool management system is to provide the right tools to the right machines at the right times. Proper tool management strategy must contain elements that allow for monitoring and control of tooling used in the production facility. Along with process planning, shop floor scheduling, tooling database, tool inventory, and tool identification, the machine tool controller is an essential hub in the tool management system. The functions of the machine tool controller related to tool management include:

- Cutter length and diameter compensation, typical of tool geometry in Figure 4.
- On-machine tool identification
- Tool breakage detection
- Tool wear monitoring
- Tool location tracking
- Tool data storage
- Tool assembly storage control
- Tool data transfer

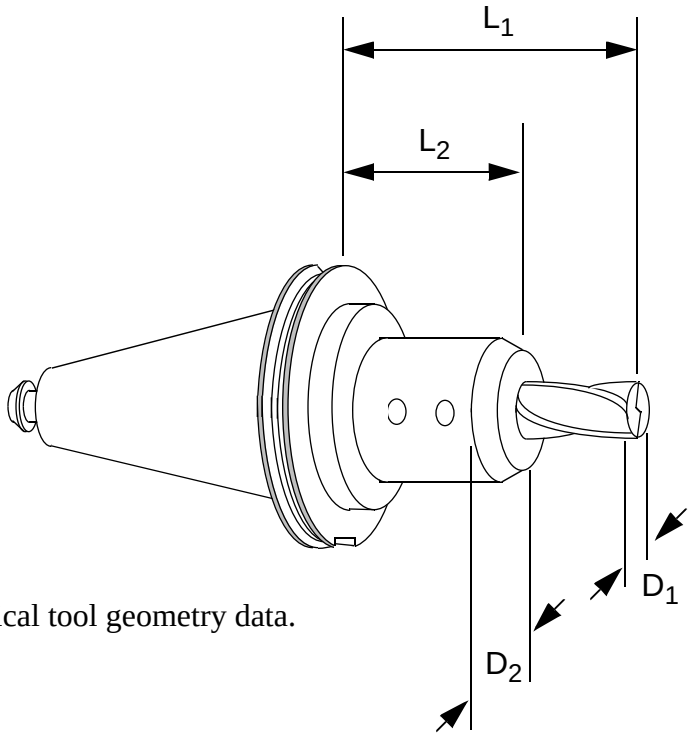

Figure 4. Typical tool geometry data.

The machine tool is the ultimate consumer of tooling and tooling data. Machining activities also generate tooling data which impacts other production control systems. This information must be accurately processed and maintained. The machine tool controller is the front line system responsible for processing and collecting data required for machining operations. This data includes:

- Accumulated cutting time
- Remaining tool life
- Magazine storage location
- Speed and feed override
- Cutter length
- Cutter diameter
- Cutter compensation values
- Tool number
- Tool assembly ID number
- Tool assembly size code
- Job number

Many tool management systems are commercially available. Many machine tool control vendors offer tool management software as an option for their controller. Almost all of these systems are completely incompatible. Unless a user selects all his systems from one vendor, systems integration is extremely difficult. Communications and data protocols are incompatible. The data types themselves may not match (e.g., no tool assembly ID number field). An open architecture machine control system would make it possible to integrate tool management applications.

NIST has implemented two components of a tool management system on the Laboratory Development Controller: an interface to a commercial radio-frequency tool tag system, and a tooling database which contains the data for the tools in the Monarch VMC-75 tool carousel. These components provide the functions necessary for applications running on the controller to verify tools and update the tooling information, most notably the NC code interpreter.

5.3 Alternate Part Programming Languages

Many companies use cutter location (CL) data output from CAD/CAM systems as a source programming language for machine tool control. The CL data must be postprocessed, or translated, into machine language for a specific controller before it can be executed. If the job has to be transferred to a different machine, the program must be postprocessed again into the machine language for the new machine. The two predominant machine languages are specified by the Engineering Institute of America (EIA) [2] and the International Standards Institute (ISO). In US companies, most controllers use the G-codes and M-functions of the EIA specification. These controllers, however, differ widely in their use of unspecified codes for functions common to a particular class of machine. Vendor variations within the “standard” programming languages have resulting in a profusion of dialects.

Translating the source programming language into the various dialects can become a time consuming task. Developing and maintaining a postprocessor for each controller dialect can become extremely expensive. Programming all jobs with one language would allow program interchange

between similar machines, reduce postprocessor development maintenance, reduce training, and ease program archiving.

To reduce the amount of resources devoted to postprocessor development and maintenance, many companies are shifting to generic post processors. With a generic postprocessor, the required parameters for a machine and controller are contained in a single data file. Rather than a different post processor for each machine, a single post processor is used and a data file is developed for each machine. To eliminate reprogramming, a standard CL file format is required. An existing standard, EIA-RS494 (Binary Cutter Location, or BCL) [6] specifies an NC file format that different-make machines can share. Standard controller input allows schedulers and shop supervisors to move jobs between similar NC machine tools without incurring reprogramming delays. Users can move jobs to remote facilities or suppliers without reprogramming.

The higher cost of BCL-compatible controllers makes users reluctant to investigate the benefits of a standard language capability. An open architecture control would provide an opportunity to implement an on-machine language translation capability at a low cost, since the entire NC code interpretation process is contained with the plan interpretation component.

NIST has investigated integrating a commercial BCL system on the Laboratory Development Controller, for both off-line program translation and on-line interpretation. With off-line translation, the BCL file is first translated into EIA-274-D, and the resulting program is run on the controller. This method is applicable to most controllers today, but suffers the drawbacks that program single-stepping or line edited are done in the translated EIA-274-D and not the original BCL. An alternative is to modify the part program interpreter so that it can execute BCL directly. This requires that the BCL interpreter provide tools for linking the appropriate function calls to the axis and I/O systems to the interpreted BCL statements, and that the controller provide access to these function calls. Alternatively, if the source code to the interpreter is provided, any necessary modifications to the interpreter can be made directly. NIST has selected to pursue the second option, executing a non-disclosure agreement with the vendor.

5.4 Spline-based motion

Tool paths that are not lines, circular arcs, or helical arcs are typically approximated with line segments. This approximation produces a small error between the desired path and the commanded path, which is termed *chordal deviation*. Non-uniform rational B-splines (NURBS) have long been favored in CAD systems because they offer an exact, uniform representation of a wide variety of common curves such as circles, parabolas, ellipses, lines, and hyperbolas [11, 12]. NURBS are also computationally efficient and allow great flexibility in defining free-form curves and surfaces. By basing the tool path on NURBS, the position error generated by approximating curves with straight lines is reduced significantly. Also, velocity errors created by commanded tool positions are significantly smaller with NURBS-based trajectories than from those with a linear approximation.

Another benefit of using NURBS is that the amount of space occupied by part programs and the time taken to execute them can be significantly reduced. Large sections of part programs often consist of many small linear moves. Each of these moves must be read by the controller and executed. In some cases, the feed rate of the machine is limited by the rate at which these lines can be read. Additionally, these part programs consume large amounts of mass storage, and can take considerable time to download to the controller.

The reader familiar with drawing programs may have already used splines: many drawing programs provide a polygon tool which creates closed or open polygons that can be smoothed. With a relatively small number of polygon vertices ("control points"), complex shapes can be created. NURBS are an extension of these smoothed curves which give greater control over the degree to which the curve approaches these control points, by introducing a "knot vector" of coefficients for the polynomial used to fit the control points.

In Figure 5a, a typical second-order curve has been approximated with the line segments AB

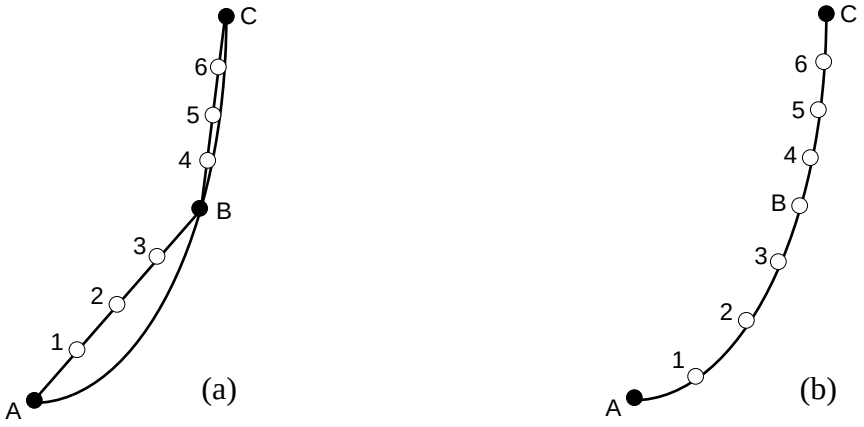

Figure 5. (a). Approximation of curve with line segments. Numbered points are generated by interpolation during machining.
(b). Fit of curve with NURBS. Numbered points and point B are generated during machining, and exactly lie on the curve.

and BC. The controller will interpolate linearly between these points during machining, generating points 1 through 6, which do not lie on the curve. This is the chordal deviation, which is usually a parameter specified to the CAD/CAM system for use in determining the number of lines to use for the approximation. The smaller the acceptable chordal deviation, the more line segments used to approximate the curve, and the larger the part program.

In contrast, if the controller were able to handle NURBS directly, the interpolated points would lie exactly on the second-order curve, as shown in Figure 5b. Also, only one motion instruction is required: a single NURBS move from A to C. The benefit is a more exact fit of the tool path the curve, with a reduction in the size of the part program (in this example, from 2 linear moves to 1 NURBS move.) Note that the interpolated points will be sent to the servo control module, which generally performs its own subinterpolation between these. The subinterpolated points are equally spaced in time, not distance, and must be computed at the servo update rate. The errors associated with subinterpolation are much smaller than those due to linear approximation in the trajectory generator, and may be minimized using techniques in [13].

The benefits are not without cost, for the trajectory generation component of the controller must be improved to handle the new NURBS motion type. While most probably this requires a software modification, the increased complexity may also require that faster computing processors be used. However, with a modular open architecture, the decision to incorporate NURBS trajectories can be

based on the application's demands, and the effects of the choice are localized to a small portion of the controller.

NIST has integrated NURBS trajectory generation in the Laboratory Development Controller. Incorporating NURBS requires the extension of three components: workstation planning, plan interpretation, and trajectory generation. The workstation planner, in this case a CAD/CAM system, must be able to represent geometries using NURBS. This is the original source of the NURBS data. In fact, many CAD/CAM systems have this capability already [14]. Since the output of CAD/CAM systems is an NC part program, new instructions in the part programming language need to be added in order to signify the new motion type. In the EMC, the part programming language is EIA-274-D, and the extensions required to support NURBS paths are summarized below:

Table 1: NURBS Extensions to EIA-274-D

G code	Function
G81	Set knot vector
G82	Define a control point
G83	Begin machining

5.5 On-machine Inspection

In an earlier effort related to the EMC, researchers from the New York University Courant Institute of Mathematical Sciences developed an open-architecture machine tool controller to serve as a platform for research on fundamental issues in chip mechanics, surface finish quality, sensor-based machining, and adaptive control [15]. This controller, MOSAIC, was based on off-the-shelf motion hardware, CPUs, buses, and workstations, and supported C-language programming in Unix environments. One application demonstrated by the MOSAIC program was automatically determining the location of arbitrarily positioned workpieces. The application was based on a mesh routing algorithm which drove a touch probe around the workpiece, generating accurate coordinates of the boundary points. Once the boundary coordinates were acquired, they were statistically fit to a model of the part, determining its "best fit" position and orientation. This eliminated the need for an operator to accurately fixture the part, speeding up the machining setup cycle.

In cooperation with the NYU program, NIST obtained the mesh routing algorithm software to demonstrate that applications such as automatic part location could be easily integrated into the EMC architecture. An analysis of the algorithm has led to some observations on the requirements for openness. These are discussed below.

5.5.1 Plan Interpretation

Because the software has been written modularly, the fundamental mesh routing algorithm required no modification. However, since the mesh routing algorithm cannot be written as an EIA-274-D program, the plan interpreter requires some extension to handle instructions signifying the initial conditions and other flags that govern the probing. This is analogous to adding a "canned cycle" for probing.

The difference is that canned cycles are macros, simply a shorthand for a fixed sequence of NC code instructions. For probing, the sequence of movements is not fixed in advance, but depend on the part geometry. Now, instead of simply parsing the instructions, expanding macros, and issuing

messages to the trajectory and discrete I/O components, the plan interpreter must handle conditional expressions that determine how the probing is proceeds as data is acquired. However, the program for this has already been written by NYU researchers, and extending the plan interpreter simply becomes a matter of linking a call to the NYU mesh routing function with the new instruction for probing, and replacing a handful of function calls in the NYU code with message passing statements to the trajectory generator.

5.5.2 Trajectory Generation

The impact of probing on the trajectory generator is minimal. Probing requires that a new motion type be added to the repertoire of the trajectory generator, implying extensibility. This motion type is a "guarded move," which is a conventional move to a particular point with the addition that a probe trip causes an immediate deceleration to a stop. A request for this motion type is generated by the plan interpreter during the execution of the mesh routing algorithm. Additionally, a flag indicating whether the probe tripped or the motion completed with no contact must be returned to the plan interpreter.

5.5.3 Servo Control

For accuracy, it is required that the signal generated by the probe cause a capture of the axis positions a quickly as possible. Any delay between the generation of the probe signal and the position capture will result in inaccurate position reporting. For this reason, hardware latching is often preferred, since the delay results in negligible probing error even at high speeds. Latching in software places a burden on the coupling of the probe signal with the position capture function. It is also desirable to trip the probe during a constant velocity portion of the move, so that the distance traveled during the delay can be characterized. The real burden is placed on the operating system of the controller, so that the function which connects the probe signal with the position and velocity data executes deterministically. In practice, this is effected by a device driver and interrupt service routine, if they can be made to execute deterministically and at high speed.

Currently, NIST has developed the software which reports the hardware position capture on the Shop Floor Controller, based on the signal from a touch-trigger probe. Software has also been developed which backs off the axes appropriately once a trigger has occurred.

5.6 Thermal-geometric error compensation

A significant source of error in the overall tool position for machine tools originates in thermal growth. At NIST, researchers have developed a method to characterize the effects of thermal growth which relies on comprehensive kinematic and geometric-thermal models of machines [16]. These models are derived from the structure of the machine and pre-process characterization measurements. The geometric-thermal model relies on measurements of actual machine temperature during machining, read from thermocouples mounted at key points on the structure. Using the set of thermocouple measurements, and the current position of the axes, the models are evaluated to produce a set of values, one for each axis, that represents the amount each axis needs to be offset in order to compensate for the thermal deviation.

The Monarch vertical machining center at NIST has been fitted with thermocouples, and is serving as the testbed for thermal-geometric error compensation integration. The mechanism by which the corrected tool offsets are integrated into the controller is based on the openness of the servo control module. First, the current position of each axis must be available to the system which evaluates the models. This exemplifies one aspect of openness: the ability of independent parties

to observe the state of the controller. In the Shop Floor Controller, the current axis positions are available through dual-ported RAM, accessible to any program running on the host computer. To support extensibility, the servo control module provides a matrix which is referenced during each servo calculation. This matrix is a homogeneous transform which contains the offsets that are to be used to modify the outputs to each axis. Initially zero, they may be loaded with offsets which originate from an external source, in this case the results of the calculation of the thermal-geometric error compensation model. This exemplifies another aspect of openness: the ability of independent parties to modify the state of the controller, to extend the functionality of the machine.

NIST is currently integrating thermal-geometric error compensation on the Shop Floor Controller. To date, thermocouples have been installed, models have been developed, and a software interface to the thermocouples have been written.

6. SUMMARY

The NIST Enhanced Machine Controller program has demonstrated the feasibility of open architectures for machine control. By assembling controllers from commercial off-the-shelf components, and integrating a variety of enhancements, the benefits of reduced development cost and improved performance have been realized. Two controller testbeds have been implemented, a Shop Floor Controller for a 3-$^1/_2$ axis vertical machining center with a tool changer, and a Laboratory Development Controller for a 3-axis desktop milling machine. The focus of the Shop Floor Controller is integrating technology and algorithms for actual machining processes, while the focus of the Laboratory Development Controller is on the software engineering required to develop standard interfaces which fit into an architectural framework. Several enhancements have been investigated and are currently under development: selectable look-and-feel, tool management, alternate part programming languages, spline-based motion, on-machine inspection, and thermal-geometric error compensation.

7. REFERENCES

1. Frederick M. Proctor and John Michaloski, “Enhanced Machine Controller Architecture Overview,” to be published as a NIST Internal Report.
2. Electronic Industries Association, “Interchangeable Variable Block Data Format for Positioning, Contouring, and Contouring/Positioning Numerically Controlled Machines,” EIA Standard EIA-274-D, February 1979.
3. Albus, J. S., “RCS: A Reference Model Architecture for Intelligent Control,” *IEEE Journal on Computer Architectures for Intelligent Machines*, May 1992.
4. Albus, J. S., Lumia, R., Fiala, J. C., and Wavering, A. J., “NASREM: The NASA/NBS Standard Reference Model for Telerobot Control System Architecture,” *Proceedings of the 20th International Symposium on Industrial Robots*, Tokyo, Japan, October 4-6, 1989.
5. Simpson, J., Hocken, R., and Albus, J., “The Automated Manufacturing Research Facility of the National Bureau of Standards,” Journal of Manufacturing Systems 1 (1), 1983.
6. Electronic Industries Association, “32 Bit Binary CL (BCL) and 7 Bit ASCII CL (ACL) Input Format for Numerically Controlled Machines,” EIA Standard Proposal No. 2918, February 22, 1992.

7. Brad Damazo, Charles Yang, and Bob Gavin, “Enhanced Machine Controller Project: Shop Floor Controller Implementation Guidelines,” to be published as a NIST Internal Report.

8. Editors, *Modern Machine Shop*, “NC Tool Management,” *Modern Machine Shop 1993 CNC Software Guide*, April 1993, pp. 187-208.

9. William A. Gruver and Mark T. Senninger, “Tooling Management in an FMS,” *Mechanical Engineering*, March 1990, pp. 40-44.

10. Ali S. Kiran and Richard J. Krason, “Automated Tooling in a Flexible Manufacturing System,” *Industrial Engineering*, April 1988, pp. 52-57.

11. P. E. Koch and K. Wang. “The Introduction of B-splines to Trajectory Planning for Robot Manipulators,” *Modeling, Identification And Control*, Vol. 9, No. 2, 1988, pp. 69-80.

12. W. Boehm, G. Farin, and J. Kahmann, “A Survey of Curve and Surface Methods in CAGD,” *Computer Aided Geometric Design*, Vol. 1, No. 1, 1984, pp. 1-60.

13. Curtis S. Wilson, “How Close Do You Have to Specify Points in a Contouring Application?,” Delta Tau Data Systems, 9036 Winnetka Street, Northridge, CA 91324.

14. Lynn Hock and Jim Hock, “NURBS is Not NURBS,” *Modern Machine Shop 1993 CNC Software Guide*, April 1993, pp. 87-94.

15. Steven Ashley, “A Mosaic for Machine Tools,” *Mechanical Engineering*, September 1990, pp. 38-43.

16. M.A. Donmez, D. S. Blomquist, R. J. Hocken, C. R. Liu, and M. M. Barash, “A General Methodology for Machine Tool Accuracy Enhancement by Error Compensation,” Precision Engineering, Publication No. 0141-6359/86/040187-10, 1986.