"Motion Server" - A Next Generation Motion Controller Architecture

J. Randolph Andrews
Douloi Automation, Inc.
740 Camden Avenue Suite A1
Campbell, CA 95008-4102
(408) 374-6322

Paper Presented at the
1996 Incremental Motion
Control Systems and
Devices Symposium

Introduction

Automatic machines using motion control grow more sophisticated. Advances in electronics allow motion controllers to more accurately position motors. However the key issue in completing an automated machine is the software which manages information and control.

Setup software helps configure general purpose motion control products for their particular use in a machine. Test software moves different sections of a machine in test patterns to confirm proper operation of the control system and mechanism. Application software describes the machine's overall behavior and provides an operator interface to direct the machine. Diagnostic software studies the machine during normal operation and collects information to direct corrective action.

Because machines are physical in nature, the software that controls them needs to have a strong sense of time. This "hard-real-time" software needs to be simple to describe and construct.

A hardware/software motion controller architecture called "Motion Server" is presented which responds to these issues.

Why isn't machine control easy?

It's unusual for an advanced motion controlled machine to be finished on schedule. Why is this so? Automation projects are by nature interdisciplinary and often have new mechanical elements, new electronics, new wiring, and new software. When problems are encountered there are many possible explanations ranging from mechanical binding to an optical sensor being out of adjustment. In many cases the total control system is composed of pieces having different commands, languages, and communication standards. This situation is pictured in Figure 1.

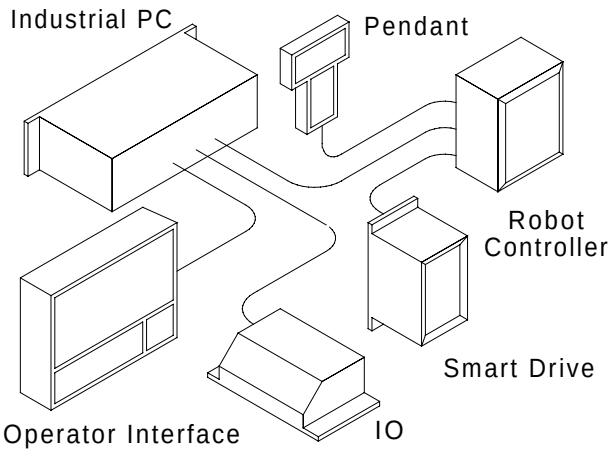

Figure 1. "Glue Engineering" Problem

Integrating these different elements together involves "glue engineering". Important information residing in each piece may not be visible through the communication links that were created during the "glue" step. Not being able to see what's going on in the pieces thwarts the system integration effort.

Machine developers need a clear view into the operation of their machine to see actual behavior and direct the machine's development.

How Can the Controller Help?

The controller should simplify communication between different parts of the system and ideally require no additional communication at all by encompassing the required control functions itself. This reduces the amount of "glue engineering" and helps maintain a simpler, more cohesive control system.

In real applications there is often an additional function required which is not part of the standard control system. A controller should accordingly support a simple, "open" hardware standard for adding these missing functions.

Software for the various setup, test, application, and diagnostic purposes should be available "off-the-shelf" as much as possible. Several of these purposes are general in nature and not application specific.

Diagnostic software should be able to tap into the control system during normal machine operation to study the machine "live". The important problems to study occur during actual machine operation, not during an artificial test case that a diagnostic tool may be limited to.

The control system should be tailorable to respond to unique application requirements. Application programs which operate at the controller sample rate allow coordination tailoring and timely data collection. As well as software flexibility the controller architecture should support hardware customization with rapid turn-around at least on a consult-the-factory basis.

Motion Server Architecture Description

Motion Server is a controller architecture involving both hardware and software elements. Figure 2 shows an overview of the control system.

The left half of the figure represents a host computer. This computer is most likely running a familiar but not necessarily real-time operating system such as Windows '95. In this computer several programs are active and communicating to Motion Server. These programs are "motion client" programs and each has its own communication channel. Inside Motion Server, indicated on the right side of the figure, are several hard-real-time programs providing services needed by these client programs. Motion Server provides real-time services on behalf of non-real-time motion client programs.

Hardware Description

In this particular embodiment, Motion Server is a long-slot ISA card containing the following items:

- 486 DX/2 or DX/4 processor
- up to 1/2 Megabyte of high-speed memory
- up to 16 axes of control
- servo or stepper control on a per-axis basis
- tandem watchdog safety system
- 48 configurable IO points
- PC/104 isolated "bridge"
- FPGA based "downloadable hardware"
- non-volatile configuration memory

Motion Server uses a single, fast, general purpose processor for all of the computation. This processor performs profiling, PID control laws, coordination, and application threads. As well as filtering mathematics, Motion Server is performing high-level language execution where the attributes of a 486 class processor are particularly beneficial. The 80 bit floating point processor in the 486 is a valuable and convenient resource when performing kinematics or specialized math functions. To keep the architecture simple and reduce communication overhead it was judged best to use the "processor budget" to purchase a single high performance processor than multiple specialized devices.

Motion Server contains 1/2 Megabyte of "cache ram" high speed memory, and no "DRAM" slow memory at all. Based on the same principle as processor selection, it was judged better to have a single, simple, high performance solution to this controller function than to support a smaller amount of high speed memory and additional slow memory. This high speed memory is specifically for the real-time

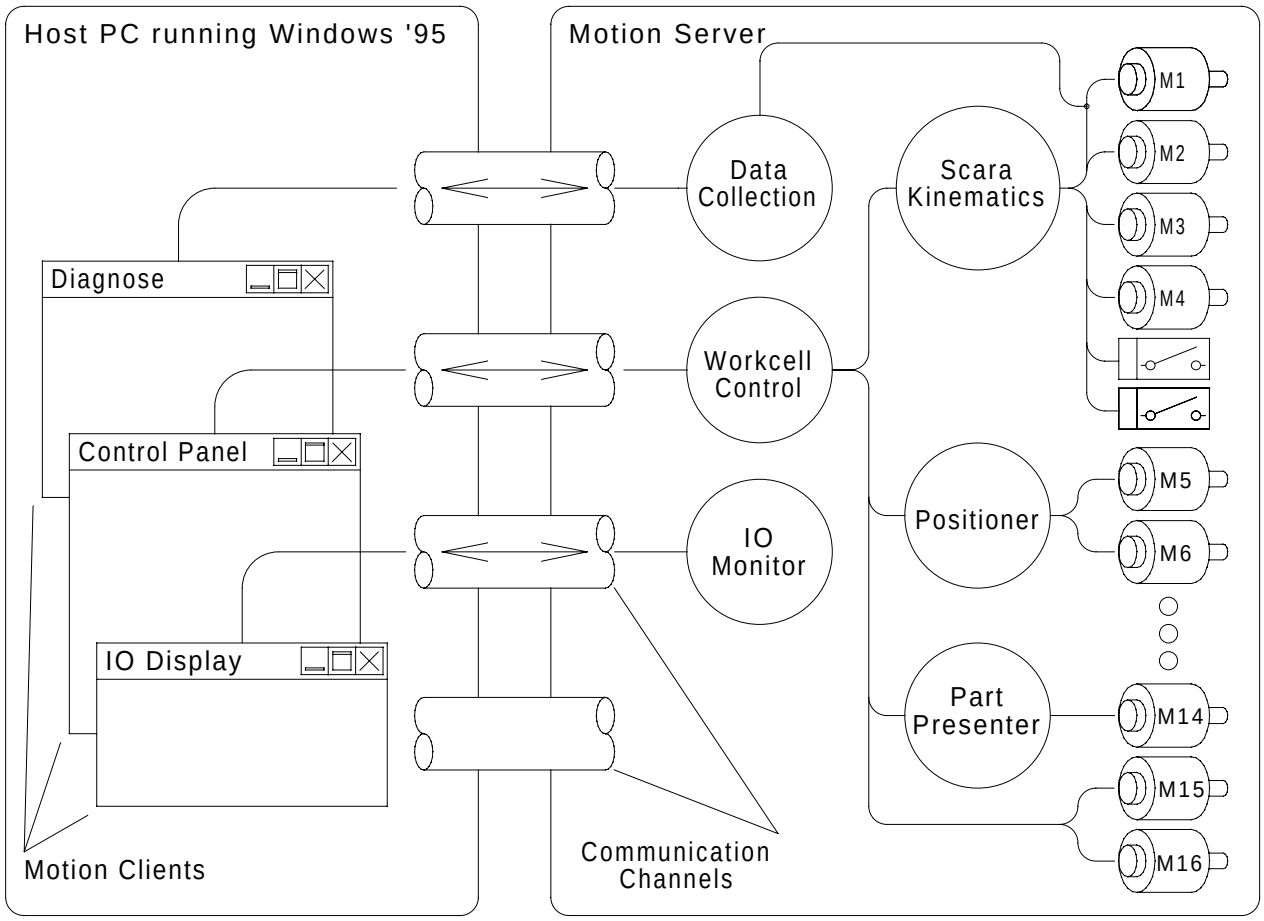

Figure 2. Motion Server Architecture

activities being performed by Motion Server. Operator control panels and software activity occurring in the host computer, on the left side of Figure 2, do not absorb any of this motion server memory.

To determine if 1/2 Megabyte of memory is sufficient a comparison can be made with Douloi's current generation controller product, SI-3000. The most memory-consuming application developed at Douloi Automation during the last 4 years with SI-3000 was a surface mount assembly machine. This machine involved 12 axes of control, 4 pickup heads, nozzle change system, synchronous part handling from part feeders of multiple types, vision system, graphical operator interface, 192 points of conventional IO, RS-485 smart IO communication, RS-232 pendant communication and multiple data bases. The real-time memory requirements of this application were 320K. 512K of memory should serve well.

Motion Server supports 16 axes of control on a single ISA card. Each axis can be configured as a servo or stepper motor axis. Each axis supports high speed capture of position, high speed compare for precise IO control, and differential encoder inputs with digital filters for improved noise immunity.

Motion Server uses a "tandem" watchdog including both an analog circuit based watchdog timer as well as digital count-down based watchdog timer. Both watchdog timers must agree that the system is properly operating for power to be applied to the motors. This tandem design safeguards the

system even if an under-voltage PC causes fundamental signals, such as the system clock, to operate incorrectly.

Motion Server contains 48 IO points which can be configured as inputs or outputs in 4 bit groups. The PC/104 isolated “bridge” allows additional digital, analog, and communication IO to be added to Motion Server. PC/104 is a popular industrial format offering good value and small size. The PC/104 functions are provided in an isolated manner so as to benefit from the “openness” of the PC/104 standard while not exposing motion server to possible problems that could occur if an add-on PC/104 card misbehaved. Even if a PC/104 card grounds its data lines, the Motion Server processor is able to continue operation and maintain control of the system.

Motion Server uses downloadable Field Programmable Gate Array technology for host communication, servo control hardware, and stepper control hardware. The motion controller hardware is "constituted" from descriptions resident in Motion Server non-volatile memory. Hardware specials, in the past requiring new board layouts and parts, can now be implemented by downloading "different hardware" into these flexible devices. This dramatically reduces the turn-around time and expense for realizing hardware specials. The "hardware" can be distributed on a floppy disk to update the non-volatile memory in the controller card.

Software Description

Motion Server provides a hard-real-time multithreading kernel which supports up to 12 threads. Each thread can execute every controller sample period. This allows application programs to describe coordination tailored relationships between axes such as robot kinematics, electronic gearing, electronic camming, tangent servo, etc. The axes in the control system can be arranged into groups which are governed by different threads. These groups can act independently or cooperatively as required by the application. Several different programs can be downloaded into the controller and operate concurrently to operate the machine and perform diagnostics.

Programming

Motion Server can be programmed in a variety of ways. A set of “personalities” is available to reduce the programmer’s learning curve. Personalities include:

- G Code
- HPGL
- Binary Command
- ASCII Command Packet
- Windows Dynamic Link Library
- Delphi VCL
- Microsoft OCX
- Douloi Object Pascal

While a particular personality is active on one communication channel other clients can communicate to Motion Server on other channels with other personalities based on whatever is most suitable for their jobs.

The different personalities available cover a range trading off ease of use with sophistication. The most advanced personality is the Object Pascal language expressed through Servo Application Workbench, a visual programming tool. SAW is specifically designed to simplify the development of hard-real-time software for motion controlled machines. SAW is similar in style to Visual Basic or Borland’s Delphi environments. The benefit of using this tool is that the outcome of the programmer’s work is tailored for real-time motion control.

As well as machine behavior, SAW can describe the operator interface and control panel of a machine. A SAW program includes in one representation the host application shown on the left side of figure 2 and the real-time behavior shown on the right side of figure 2. The communication link required for their cooperative relationship is provided automatically. A Servo Application Workbench Programming Session is shown in Figure 3.

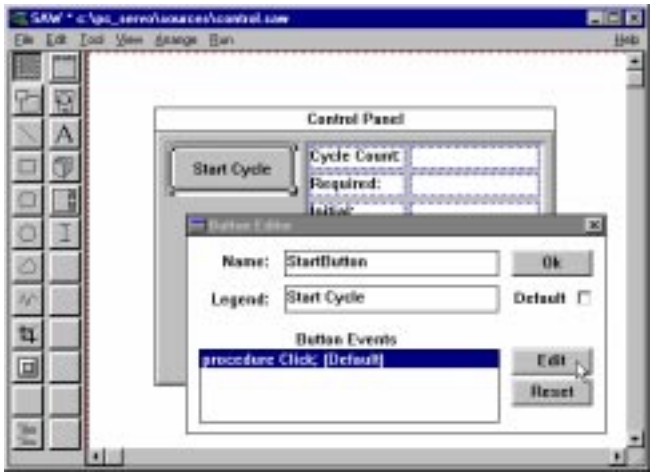

Figure 3. SAW Programming Session

An operator interface button has been chosen from among the assortment of controls on the left side of SAW. The attributes describing the button are being placed in this dialog including the buttons name, legend, and behavior when pressed.

Application Example Comparison

To illustrate how the Motion Server architecture simplifies advanced machine control a comparative example is presented.

Functional Description

Figure 4 shows an overview of a robotic workcell.

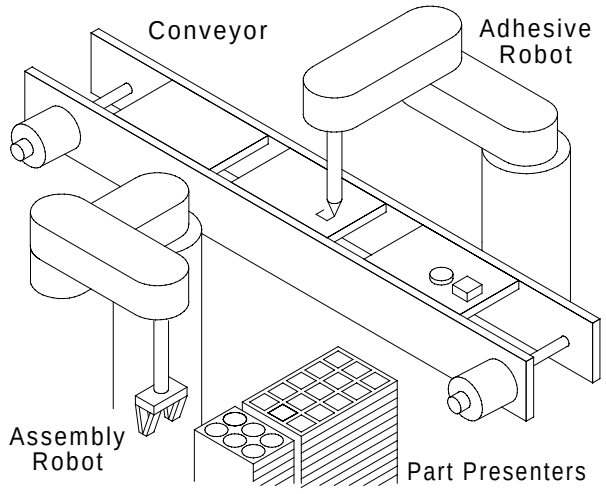

Figure 4. Assembly Workcell

The workcell contains the following elements:

(1) 4 axis scara assembly robot with 2 axis end effector
(1) 3 axis scara adhesive robot
(1) 2 axis adjustable width conveyor
(2) 1 axis part presentation elevators
(10) pneumatic actuators
(50) digital inputs
(20) digital outputs
(1) Graphical Operator Interface

Assembly bases arrive and queue on a variable width conveyor. Adjusting the conveyor width is done by stepper motors at machine setup time and is not time critical.

Bases are detected by optical sensors and fixtured by pneumatic pins. The 3 axis robot applies an adhesive pattern to several surfaces inside the base while the assembly robot retrieves 2 parts from the part presentation tools. The adhesive robot then retracts and the assembly robot places the parts. The base is then released downstream and replaced with the next incoming base.

After a part has been removed the part presentation tools use servo and pneumatic actuators to advance their next parts into position for retrieval.

Application Issues

This application involves concurrency. While the adhesive robot is performing the glue operation the assembly robot is retrieving parts. As soon as parts are retrieved, each part feeder performs the necessary actions to present the next part as the robot is moving towards the assembly locations.

Along with independent operation of these different systems is the need for proper “synchronization” at specific points. If the part presentation tools have not yet provided a part, the assembly robot must wait for one. If the adhesive robot has not completely retracted, the assembly robot must not attempt to move over the base because of the risk of collision.

The easiest way to describe this type of behavior is with simple, small programs, one for each section of the system. These small programs then communicate to each other to provide the necessary synchronization.

Conventional Approaches

A conventional approach might use pre-packaged robot controllers, one for each robot. An additional "stand-alone" multiaxis controller might be chosen to handle the robot end effectors and additional workcell tooling. A PC is used as the “workcell controller” coordinating these elements. This approach is represented in Figure 5.

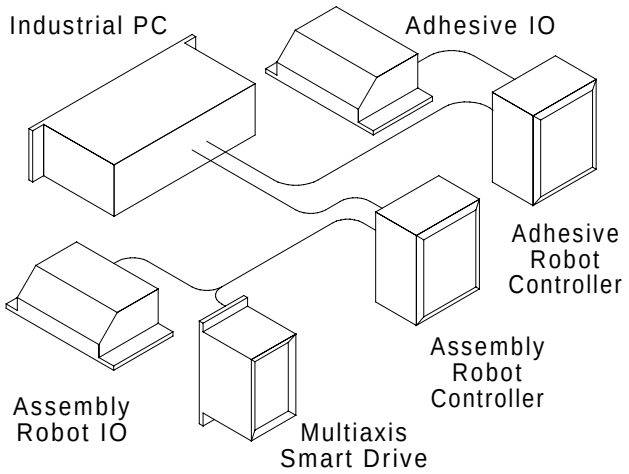

Figure 5. Conventional Control Approach

This approach does provide the benefit of having several small programs running. Each piece of equipment has its own dedicated controller running its particular program. However there is an "information visibility" issue which thwarts system integration and diagnostics.

Curtis S. Wilson of Delta Tau Data Systems in a recent article quoted the following observation regarding the expense of "Intertask communications":

The closer together two tasks are done, the less expensive the communications are between the tasks for a given bandwidth. A good rule of thumb is that for each stage more remote the communications between two tasks, the cost of the communications for equivalent bandwidth goes up by an order of magnitude. Of course, in practice, the bandwidth of the communications goes down at the more remote levels. In today's industrial electronics, the stages of interest for intertask communications are:

1) Between registers on a single integrated circuit
2) Between ICs on a single circuit board
3) Between circuit boards in a single backplane box
4) Between backplane boxes in a single enclosure
5) Between enclosures in a single room
6) More remote communications.

I would suggest that "expense" as described here might also include system integration expense. The conventional approach as described would be case 4, the robot controllers and PC in a single enclosure.

In response to this desire for close communication the point might be made "But those are independent systems. They should be separated from each other". An important principle in software engineering is "encapsulation", the deliberate concealment of information in one system from another. This important concept helps partition problems, reduce complexity, and safeguard integrity. In the abstract world of software, separate systems can be independent and truly uncoupled.

However in machine control there is physical interaction between systems creating a coupling that may need to be understood. The robot controller and end effector may be conceptually quite separate and electrically driven from different controllers. However during operation unexpected torques in the robot elbow, for example, may result from the end effector gripping a part off-center. To see and diagnose this condition involves collecting torque information from the elbow and the gripper at a rate on the order of the controller sample period. This data collection needs to occur during the actual grasping action as described by the application software. This type of real-time "information visibility" degrades with each step down the list described by Mr. Wilson. This particular data collection example becomes more difficult when the information is in different controllers.

An additional problem with the decentralized control illustrated is that the setup, test, application, and diagnostic software for the robot controller is different than for the "stand-alone" controller and requires an additional learning curve.

An alternative conventional solution is to place into the main PC controller cards for the necessary IO and motion control. The coordination between these elements could then be accomplished with a hard-real-time operating system such as QNX. This is a very capable solution however it does require extensive knowledge of hard-real-time programming techniques to make the entire system operate. Programming in QNX or other tools of similar capability is not like programming with Visual Basic.

Motion Server Approach

Figure 6 shows how the motion server architecture would apply to solving this example problem.

This approach also has several distinct "controllers" as indicated by the labeled circles. However all of these autonomous controllers are inside Motion Server as individual application threads. Motion Server takes responsibility for the real-time requirements of the application. The IO resources are selected and axes allocated in groups for use by the different application threads. Because more IO is required than the built-in 48 points of IO, an additional IO card is added to the PC/104 bridge. Each section of the machine is then directed by its own application thread to keep that part of the machine running.

Each scara robot has kinematic calculations occurring so that the mechanism can perform straight-line XY movements. These kinematic capabilities are provided by a pre-assembled software element as a standard part.

As well as the workcell control application program, several diagnostic programs are running to aid in system development.

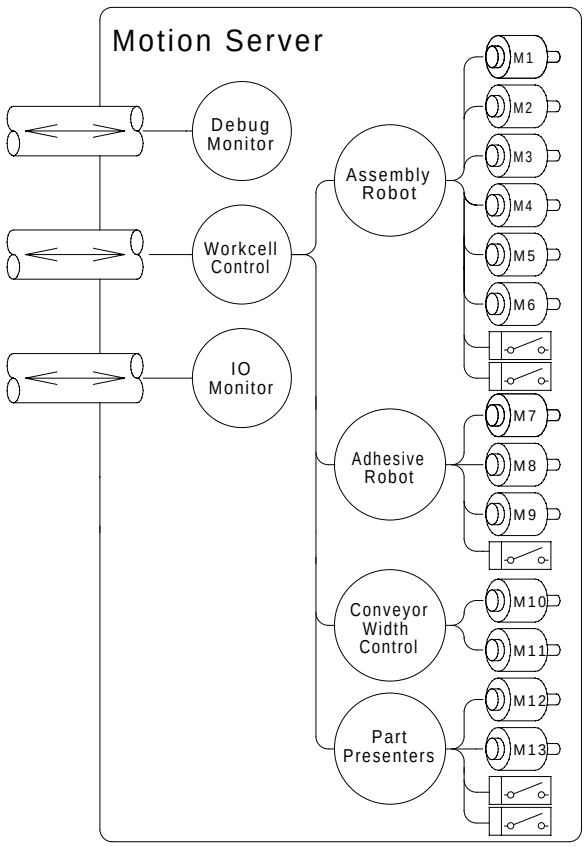

Figure 6. Motion Server Approach

Communication between threads is accomplished by simple shared program variables. Because all of the independent activities are in the same processor the communication issues are greatly simplified. This arrangement is at worst list item 2, information travelling on the same circuit board between chips (high speed memory and processor) or at best list item 1, information moving between registers (or internal cache) on a single chip.

Architecture Benefits

Simplified real-time communication

All of the real-time information in the system is in the same processor, and that processor is on the bus of a computer which can display the information. Information shared between program threads coordinates machine activities and is simple to establish.

Improved Diagnostics

The high visibility of system information allows diagnostic tools to readily access whatever information is required. Visual programming tools enable rapid construction of instruments that can ask application specific questions concerning the dynamic behavior of a machine.

High Speed Application Programs

Because application programs are compiled rather than interpreted the 486 processor can execute applications very quickly. Sample rate applications are common and enhance the controllers ability to be tailored.

Flexible Hardware

Beyond flexible application programs, the hardware itself can be reconfigured electronically to reduce development time for application specific hardware specials.

Summary

Motion control technology is becoming less about the physical dynamics of moving objects and more about the management of information. This information is managed by software which is used to setup, test, operate and diagnose advanced machines. An architecture has been presented which provides a broad set of resources, is extensible, and simplifies access to this time-critical information. The architecture accommodates a variety of pre-built tools and personalities to simplify and expedite the construction of automatic motion controlled machines.

Bibliography

1) "A Sercos Alternative", Curtis S. Wilson, *Motion Control Magazine*, page 13, January/February 1996

2) Andrews, J. Randolph: "Advanced Motion Solutions Using Simple Superposition Technique", In *Proceedings of the Twenty Third Annual Symposium on Incremental Motion Control Systems and Devices, San Jose, CA,* 1994.

3) Andrews, J. Randolph: "An Advanced Motion Control System Architecture Based on a 386 PC", In *Proceedings of the Twenty First Annual Symposium on Incremental Motion Control Systems and Devices, San Jose, CA,* 1992.

4) Cox, Brad: *Object Oriented Programming: An Evolutionary Approach*, Addison-Wesley Publishing, 1986, 1991

5) W. Brogan, *Modern Control Theory*, Prentice Hall, Englewood, New Jersey, 1991

6) Brooks, Frederick P., The Mythical Man-Month, Addison-Wesley Publishing Company, Massachusetts, 1995

7) Budd, Timothy: *An Introduction to Object-Oriented Programming* Addison-Wesley, Massachusetts, 1991

8) Ellis, George: *Control System Design Guide*, Academic Press, San Diego, 1991

9) Franklin, Gene & Powell, David: *Digital Control of Dynamic Systems*, Addison-Wesley, Massachusetts, 1981

10) Meyer, Bertrand: *Object-oriented Software Construction*, Prentice Hall, New York, 1988

About the Author

J. Randolph Andrews received his B.S. M.E. in 1981, B.S. E.E. in 1981 and M.S.M.E. in 1983 from the Massachusetts Institute of Technology.

Andrews spent 4 years at Hewlett Packard's corporate research laboratory in the Applied Physics Research Center as well as the Manufacturing Research Center.

The following 4 year period was spent with Galil Motion Control.

In July '91 Andrews founded Douloi Automation, Inc. to provide motion control hardware and software systems for use with Microsoft Windows. Douloi Automation also provides turn-key automation solutions for advanced automation applications.

Professional interests include motion control, software/electrical/mechanical system design trade-offs, high abstraction programming and visual programming techniques and tools.