AIAA-94-1185-CP
N94-30535

6030

59-63
207632
P. 10

Confessions of a Robot Lobotomist

R. Marc Gottshall - Specialist Engineer
Boeing Commercial Airplane Group
Seattle, Washington

Abstract

Since its inception, Numerically Controlled (NC) machining methods have been used throughout the aerospace industry to mill, drill, and turn complex shapes by sequentially stepping through motion programs. However, the recent demand for more precision, faster feeds, exotic sensors, and branching execution have existing Computer Numerical Control (CNC) and Distributed Numerical Control (DNC) systems running at maximum controller capacity. Typical disadvantages of current CNC's include fixed memory capacities, limited communication ports, and the use of multiple control languages. The need to tailor CNC's to meet specific applications, whether it be expanded memory, additional communications, or integrated vision, often requires replacing the original controller supplied with the commercial machine tool with a more powerful and capable system.

This paper briefly describes the process and equipment requirements for new controllers and their evolutionary implementation in an aerospace environment. The process of controller retrofit with currently available machines is examined, along with several case studies and their computational and architectural implications.

Introduction

In response to the more complex machined shapes demanded by modern aircraft, the Air Force sponsored numerically controlled milling machine research at the Massachusetts Institute of Technology's Radiation Laboratory in 1949. The fusion of the then fledgling digital computer technology with servo control techniques allowed demonstration of a prototype NC machine in 1953 [1]. Over the ensuing forty years, new CNC capabilities have dramatically enhanced the way airplanes are made. CNC computers have become smaller, faster, and cheaper; through the use of innovative sensors, automated work cells can both monitor and control production processes as well as the parts they create. Upstream systems can create and store part programs, collect and analyze process data, and monitor/diagnose individual machines. In general, the processes being performed are more complex, highly precise, intolerant of delay, and are being automated at an ever-accelerated pace.

When tailoring a controller for a machine tool application, two critical considerations must be taken into account: process complexity and life cycle cost. The desire to improve product quality and reduce manual labor has caused automated systems to become more and more sophisticated. On the control side, automation applications require ever increasing amounts of software that execute on powerful computers with extensive memory. On the process side, smart-sensor based systems provide tighter control of production monitoring, quality, and reliability by collecting massive amounts of data during process execution. This data must be organized for use by both the process control and upstream business systems. Clearly, what was once a single computer operation has now become a network of 5 to 10 intelligent computer subsystems, each of which is usually a microprocessor-based smart box. The function of each subsystem is unique yet all subsystems contribute to producing a better product.

Examples of smart-sensor based subsystems include machine vision for process inspection and statistical analysis, and thermal scanning devices to monitor material growth. Data transfer of part attributes, quantities, and messages require networking capability to disk storage, file management , and company business systems. Further complicating the automation process is the need for a host system which is flexible enough to coordinate all subsystem information and make adjustments to the process in real-time. The host must also interface hardware and software to multiple communication protocols.

Cost and Complexity

While issues regarding process complexity represent the factory side of the automation problem, the business side is concerned with controlling cost. The vast amounts of software generated for application development, programming, and software maintenance must be structured in order to control life cycle costs. Because these automation systems are multi-computer based, organizing and directing in-process information mandates complex decision making algorithms. For

example, many processes require the system to adapt to changes in the process based upon input data, factory problems, and machine interrupts. To effectively implement such complex process algorithms, application software is usually developed using structured analysis and design. Structure design tools benefit the software life cycle in development, maintenance, and documentation. However, it is not always possible to take advantage of cost savings using structured design tools unless the computer language can support such development.

Typical software maintenance costs for complex automation applications can be excessive due to the diversity of languages, controllers, and variety of processes. For example, most NC, CNC, and DNC machines utilize control language based upon ladder logic. Other languages such as Allen Bradley's Siprom are used in conjunction with ladder logic when developing a machine application. Large multi-function systems written entirely in ladder logic pose a formidable maintenance task . The maintenance problem is compounded further since robotic control systems often use custom languages (such as Karel, Rail, V+, etc.). Each of these unique languages must be supported by programming staff. Factors such as language, processing capability, interconnectivity, communications, and code reusability must be weighed against what the company can afford to spend throughout the software life cycle.

The issues of process complexity and software life cycle are interdependent in the automation environment. The interdependence can be examined by breaking down these issues into further detail. First, process complexity involves key factors such as programming, communications, data transfer, control of input/output functions, and motion control. Life cycle costs, on the other hand, involve computer languages, maintenance, training, upstream compatibility, and software reusability.

Process software can be partitioned into six distinct functional groups. Generalized categories include process control, communications, file storage and transfer, digital and analog input/output, motion control, and vision processing. Of these categories, serial communications has become a critical link for most automation applications within Boeing.

Serial Communications

Many new applications utilize microprocessor-based smart boxes which can control an entire section of a process with little intervention from a host computer. The ability to allocate tasks to multiple smart boxes reduces the work load on the main controller. In addition, it provides system modularity which can reduce factory down time and part replacement. The majority of these smart boxes provide serial ports for communication. In order to reliably communicate with multiple smart boxes, the system programmer needs to have standard serial communication functions available within the host controller's language. A set of common tools might include full ASCII character recognition, basic character input/output, and configuration of the I/O port. Advanced features include data buffering, operating system notification (via flags or interrupts), and the ability to apply protocols such as Kermit, Xmodem, etc. to data transfer. Many controllers do not allow much control over a serial port, resulting in "kludging" the existing software base to create a semi-functional communications path.

Several aerospace applications require the use of thermal scanners for monitoring temperature changes and part growth the work cell. Interfacing and manipulating the data provided from these scanners has proven to be a programming challenge. Each controller has a unique implementation of the RS-232 standard. Furthermore, some controllers use restricted data formats, which limit the flexibility of the system. Still other controllers require special manipulation of the serial port hardware to make the port functional. Consequently, special communications software must be written after the serial port has been studied through a network analyzer. Compounding the problem is the lack of an RS-232 standard on the smart device. The result is the communications software must not only conform to a non-standard format at the controller side but also on the sensor side.

Protocols such as Kermit, Xmodem etc. have been successfully used in the computer industry for years. As more embedded PC boxes sprout up in automation applications, the need for a robust communications tool set resident in both the host controller and sensor systems is continually overlooked. In addition to serial communications, smart boxes are synchronizing communication with digital I/O. End effectors and manual operator interfaces can use combinations of serial communications, discrete digital I/O, and analog input/output. End effectors can be considered as completely independent machine processes. Smart controllers are used with end effectors to control valves, drill motors and part manipulators. Here again, serial communication is used to set up the end effector and control the process in real time.

Digital I/O Control

Assembly and manufacturing applications require synchronization of multiple control relays and valves using discrete digital I/O. Process control is dependent upon the ability of a host controller to receive serial information and/or discrete digital I/O, decode the information, then make a decision affecting the next step in the process. Programmable Logic Controllers (PLC's) have been used for this task. The PLC is a cornerstone in many Boeing automation applications due to its "bulletproof" ability to control process I/O. Other benefits include a large base of people who program and trouble shoot in ladder logic.

In addition to PLC's, most control system manufacturers provide both digital and analog I/O. These I/O's are interfaced to operator control panels, process switches, valves, and a multitude of sensors and indicators. While I/O interfacing is somewhat standardized, tools for developing I/O control algorithms are not. Programming a PLC for interfacing to an operator control panel can be difficult due to the lack of a rich language base. Designing a system in which I/O's can be placed in logical groups is dependent on where the grouping takes place and how many I/O's are required.

Distributed I/O boxes aid in modularizing the system design, but also complicate the system by the sheer numbers of sensors being processed. The host controller must have intelligent control over all I/O's both in hardware and software. Many real-time processes require high speed processing of sensors in order to avoid catastrophic failure. This implies a group of dedicated high speed I/O's in addition to simple valve and switch control. The inherent nature of high speed data acquisition demands computing power as well as robust hardware. The problem is further complicated by the diversity of cables and connectors required to interface the sensors.

The basic process of reading a digital input or setting a digital output is not complex. However, when that process must be carried out at high speeds, the physics of transmission lines cannot be ignored. Further, the host controller may have to read several sensors at once, perform numerical computations on the data, iterate a decision tree, and execute a reactionary function. Adding to the myriad of hardware interfaces are the variety of timing requirements for data acquisition. Coordination of the system I/O's together with the application complexity generate huge amounts of control software.

Machine Motion

In many aerospace automation applications, the issues discussed above are secondary to precise control of machine motion. Machine motion is generally executed in joint or world coordinate systems. The dominant trajectories for machine controllers are joint or linear interpolated motion. The end result is to cause the tool tip attached to the machine to perform the required movement. NC machines utilize RS274D code to perform these movements. This standard was developed in the 1950's, before the application of matrix algebra in motion control. Today, robotic controllers use forward and inverse kinematics to drive multi-axis machines. Inverse kinematics allow the controller to compute where the tool tip is with respect to the coordinate base of the machine. This function is not possible with most NC machines.

Manufacture of aerospace grade parts demands high positioning tolerances on the part of the machine. NC machines have been capable of this for years provided the part being machined is always fixed in a specific position in the tooling jig. The NC machine can probe the part and account for offsets in the X, Y, and Z axes but it cannot adjust for changes in yaw, pitch, and roll. Preparation and assembly of parts such as fuselage panels involve path motion and positioning along complex contours. (This type of operation requires machines with 5 to 6 axes of motion.)

An NC controller can be programmed for complex motion but cannot adaptively adjust during the process. This is because RS274D code being executed by the machine is spatially fixed to either the machine or the part reference frame. Thermal growth affects machining tolerances due to the large size of many aerospace parts. The part, the tooling fixture, and the machine bed are subject to different growth fluctuations due to the materials they are built from. The goal is to produce a part with very high machining tolerances yet an NC machine cannot fully adapt to the dynamic growth changes caused by thermal effects. Controlling motion using kinematics has a distinct advantage by being able to dynamically create new frames of reference.

The part program is spatially fixed but a robotic controller can establish an offset reference frame in world coordinates using probing techniques. This reference frame can be used to transform the original part data to fit the current orientation of the part and tooling jig. Other processes require drilling of holes normal to the part surface. The normal vector and position must be computed just prior to drilling the hole. Again, this is not possible without the use of kinematics to locate the tool

tip relative to the part. These operations require more computing power from the machine controller as well as the ability to store and transfer data generated by establishing in-process reference frames.

Language and Compatibility

Transferal of process data leads into the area of company business systems. The issue of upstream compatibility relates to the machine controller communicating through an established network protocol to a company data base. Unfortunately, upstream communications is tightly intertwined with the language used by the machine controller. Some systems use a server type architecture for communicating to the company database. This allows greater flexibility when changes are made to the system but the machine controller must still provide process information to some other computer based system. The focus of the next section is what role the machine controller language plays in interfacing not only to a server system but more importantly to the application itself.

The computer language of a control system plays the executive role in "gluing" application subsystems together. The language must provide a rich set of functions including input/output, file management, mathematical, decision iteratives, and graphics. Another important feature of the controller language is its ability to reflect the language syntax as readable structured text. It is extremely beneficial to be able to define and name software variables using meaningful words. Moreover, the extent to which the language lends itself to structured analysis and design implementations has far reaching impacts on costs incurred during the software life cycle. Automation software development, modification, and maintenance is a costly process within the Boeing company.

Utilizing multiple languages for an application has several drawbacks. Many companies worldwide use ladder logic as the standard for developing, implementing, and debugging sequential steps in automation and machining applications. Although newer languages may be far simpler to understand, an enormous base of people trained in ladder logic already exists. Reeducating such a large and sometimes unwilling work force is an immeasurable task.

Manufacturing companies have significant investments in existing machinery. Coupled to the machinery are support staff to maintain, operate, and reprogram production applications. Training for most of these companies is not economical. In addition, the choice of which control system and which language to standardize on is continually evolving.

Standardization of a subset of languages for applications is nearly impossible. Each automation application has specific requirements. These requirements cannot always be met using one manufacturers control system. A new system which fits the application may be purchased. This usually means a new control language with a different set of operating attributes and characteristics. Programming for the application now requires a "learning curve" with the new language, thus adding to software life cycle costs.

The variety of control systems, PLC's, and motion control cards used within Boeing are tied directly to the number of languages requiring maintenance and support. Each manufacturer has the "best" language for their machinery. Thus, every machine has one or more programming "specialists" intimate with that machine's language. Many of these machines have restricted language functionality.

Aerospace assembly applications require changes and modifications to the software as improvements are made in the process. When a controller with restricted language and/or functionality is used, the controller manufacturer must supply any customized software routines. These unique software requirements can add as much as 50% to the cost of the controller. Another cost burden is the lack of reusability of process code.

A company may expend considerable sums on in-house and customized software which cannot be transferred to any other controller. Most code developed for PLC's is application specific and cannot be migrated to future applications. In addition to the PLC, the controller language may not be portable to a similar controller. These issues pose a formidable argument for finding a single portable robust language for the entire application.

The diversity of applications within Boeing does not allow for standardizing on a single language or controller. However, a controller with a robust language function base allows for immediate application of skills used with other computer programming languages such as Basic, C, Fortran, and Pascal. Computing iteratives such as FOR, IF--THEN, WHILE, DO and CASE provide high level syntax necessary for control of complex processes. These factors are sought after because they greatly reduce the maintenance costs by providing a common set of characteristics already understood by computer programmers. Another area of concern involves connection through a network to company

business systems and storage facilities. The vast amounts of process data being collected and analyzed by upstream systems is transferred using many different network protocols. To provide this function, a controller or host computer must have memory for file storage and control of one or more protocols for file uploading and downloading. Some applications require data transfer using custom protocols developed with the controller language. Many of the older control systems support the crudest of data input and output. This can slow the automation process and also affect overall production costs. The number of process and upstream computer systems involved in the automation process continues to grow resulting in increased layers of software. The software development environment for each layer affects the overall time to production. Software development for the machine controller involves several phases.

After a structured design has been developed, the initial coding phase of all machine functions takes place. Following this phase is test and modification of the software with or without the machine in the loop. At this phase, all subsystem software is individually tested. Integration phase involves debugging all subsystems together with the machine controller. Once the subsystems are connected, all languages must be able to communicate through the main controller. The debugging environment on the controller now becomes a critical tool in testing the system operation.

Multiple modifications to the application software are made by the system programmer during this phase. Continual updating of the application software can be very time consuming depending upon the efficiency of the debugging and programming environments. For example, a compiler based language may be more powerful in terms of functional capability yet continuous compilation, linking and perhaps downloading can be extremely time consuming. On the other hand, an interpretive language can be immediately modified and tested without compilation, or linking. At this point, the use of one language for all subsystems can significantly reduce the programming complexity as well as the manpower required to get the application on-line.

An area often overlooked during this phase of software development is the end user or factory operator. While the efficiency of the development environment plays a significant role in bringing the process on-line, it must also provide a rich graphical user interface (GUI.) Most aerospace automation applications require one or more operators in the loop to monitor the process. The simplicity with which the process can be graphically represented to the operator insures better participation during part manufacturing. An efficient debugging environment for graphic objects such as icons which activate process functions is not available on many control systems.

Once these development phases are complete and the application is on-line, the software maintenance phase is activated. Inevitably, the process requirements change as the product is improved. Modification forces changes in the application software and usually reprogramming of some of the process programs. Here again, the development environment is critical to making rapid changes in the process. A system which supports off-line development and test can be extremely cost effective in the factory environment. Conversely, stopping production to modify and test application code can be costly.

Control System Requirements

The issues of process complexity, control system and language, life cycle costs, and previously successful projects are considered during the planning and design of an automation application. Because of the complexity of aerospace manufacturing, the control system is usually the host in orchestrating a process. There are many simple operations being performed at Boeing requiring PLC's and/or rudimentary control systems. The wide range of complexities of applications forces Boeing to choose different controllers for different applications. Alternatively, standardization of control systems would reduce the level of automation manufacturing by limiting applications to the technological capabilities of the control system.

Advanced applications may require a system which controls 1 or more multi-axis robots and several dependent/independent axes of motion. Dynamic coupling of axes in some applications may also be a requirement. Simultaneous control of serial communications and digital I/O information may be essential. Advanced applications may use machine vision for inspection or vision guided motion. Moreover, a prioritized response to critical interrupts during process execution is usually mandatory. These pre-requisites place a formidable load on any controller.

Factors such as multi-tasking capability, task prioritization, and time slice assignment become fundamental criteria for the controller's operating system. Without these capabilities, the control system cannot effectively perform complex automation tasks. In addition to operating system performance is the efficiency and reliability of internal coupling between hardware and software in a machine controller. The

operating system running underneath the language is usually hard coded to motion control boards, digital I/O interfaces, hard disks, and emergency stop circuitry. Multiple microprocessor systems controlling trajectory generation, digital closed loop servo control, external communications, graphics, vision, and power management are all interdependent.

The application complexity determines which of these factors are required to implement the process. Another consideration in controller selection is the number of axes and type of motion required. An application may not have any motion control or it may be a multi-axis machine with vision guided motion. This implies two controllers with very different sets of functional criteria. Thus another key factor in controller selection is the configurability of the system. A control system which can accept a number of optional subsystems to meet different requirements provides a cost effective application solution.

Collecting the topics and issues discussed in this paper provides a general outline of problems which exist in the manufacturing environment. There are still more problems and new solutions being developed today in factories around the world. This paper is not intended to be a catch all of automation issues, but an insight into the growing complexity of factory automation. The next section discusses four case studies of systems currently in use within the Boeing company. The general system block diagrams are presented with a discussion of some problems and solutions related to each system. The exact details of the application are omitted in order to protect any proprietary information.

Case Studies

System 1

System 1 uses an Allen Bradley 9/260 series controller to perform processes on stringers and stringer clips. The system executes RS274D coded programs and controls two axes of motion using incremental encoder feedback for positioning. Figure 1 depicts the hardware block diagram for this system.

The operator control panel is part of the control system. This controller has two serial communications port,: one for DNC downloading of part programs from a file server, the other retrieves data from a thermal scanner. A specific DNC protocol had to be adhered to in order to

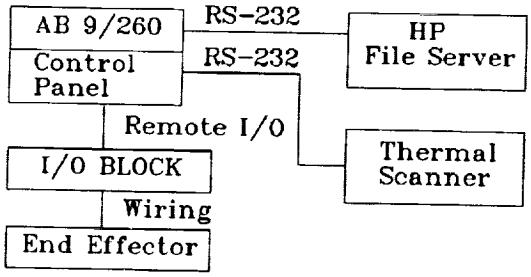

Figure 1

Number of axes	2
Number of I/O's	60-70
Number of serial ports	2
End Effector	1
Languages	PAL, SIPROM
Lines of code	800-1000

Table 1

transfer program files. A network analyzer was used to re document and debug the transfer protocol. Only one RS-232 port can be used at a time, as the second port is not a fully functional RS-232 port. The communications protocol is specific to AB. Different ASCII characters sent to this port cause predefined functions to occur. Thus, the limitations of the communications set reduced the overall flexibility of the system while increasing development time.

The application language for this system is ladder logic (PAL). The development environment consisted of separate software packages provided by Allen Bradley. PAL code was developed off-line on a PC using an AB editor package. The software was then downloaded to the AB 9/260. Debugging was accomplished by running the PAL programs while monitoring the process on a remote PC. The application code could not be single stepped for debugging. The monitor process can be started and stopped only. Motion parameters include: gains for P, I, & D, gain break-point parameter, following error limit. There are no pole or zero adjustments for the digital closed loop servo control.

System 2

System 2 uses conventional cutters mounted in an electric router to trim the periphery of composite parts for aircraft. Figure 2 depicts the hardware block diagram for this system. The part periphery are defined to tooling edges where a robot slides a router bushing. This system uses a CimCorp CimRoc4000 controller to perform all robot motions. In addition, the router motors have controllers to perform all router sensing and control of the electric routers. Material handling shuttle tables are controlled by PLC's based on digital signals from the robot controller. There are over 128 digital input and output points defined and three serial ports for the printer, router controller and position probe. Software was developed in C, running under DOS.

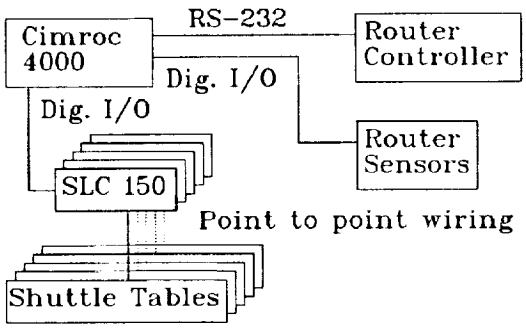

Figure 2

Number of axes	7
Number of I/O's	200
Number of serial ports	1
End Effector	Multiple
Languages	C
Lines of code	30,000+

Table 2

The DOS/C development environment made use of existing skills to efficiently implement a number of operator security functions. Graphical user interfaces were developed with the aid of a commercial graphics package and libraries for serial communications and ISAM databases were used extensively.

The most severe limitations were associated with the use of a single tasking operating system (DOS). Minor difficulties were encountered with network communications owing to interrupt collisions between the network card and communication cards needed to direct the motion control cards in the real-time back plane.

System 3

System 3 utilizes an AB 8600 controller interfaced to a 7-axis JOBS Jomach 16. This controller manipulates the Jomach 16 as well as various end effectors used in fuselage assembly processes. Figure 3 depicts the hardware block diagram for this system.This application also uses 3 PLC's, one for interfacing to a tool storage/retrieval rack, and two others for controlling the position of tooling headers. All three PLC's are connected to a host AB8600 using "Data Highway". Each of the PLC's uses "Remote I/O" for inter-PLC communication.

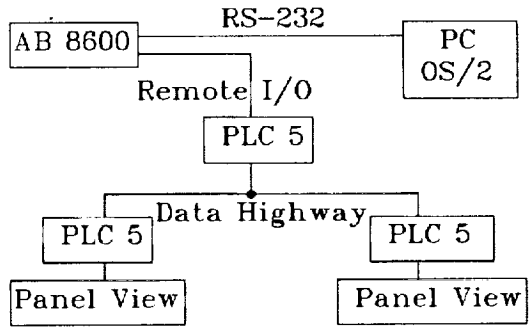

Figure 3

Number of axes	9
Number of I/O's	200
Number of serial ports	1
End Effector	Multiple
Languages	PAL,SIPROM
Lines of code	6,000+

Table 3

This system required dynamic coupling of axes during end effector drop-off and pick-up. The controller provided this capability through hardware partitioning of the axes. Memory on the 8600 CPU was also partitioned and used for up to 5 different tasks. Dynamically coupled motion was achieved using Allen Bradley's Axis Manager software.

The complexity of the application required the use of PLC's in addition to the system digital I/O blocks. Because of the difficulty in programming the PLC

interface with the operator console, two Allen Bradley "Panel View" systems were used. The PLC's use "Remote I/O" to communicate with the operator consoles, and discrete I/O to activate motion control cards.

The system integrator used Siprom and ladder logic languages to implement all process functions. The serial communications protocol used by the AB8600 is specific to the controller, and it was necessary to use a network analyzer to determine how to implement reliable communications with the AB8600. Programming tools for graphics display were inflexible and poorly documented. All GUI's and interfaces with the 8600 CPU card cage were controlled via a PC.

Since the response times for probe contact were inconsistent, programming custom probe routines for probing normal to a surface was particularly difficult. Machine motion in some applications was not as smooth as expected, due to the length of the SIPROM code and the loop execution time.

To interface a thermal scanner in this system required a usable serial port. Further, coding of customized M-codes routines in SIPROM were required for retrieving and computing the thermal data.

File operations have some minor restrictions. Downloading of files is limited to 6 ASCII characters for file names. Formatted file lengths are limited to 255 records (132 characters per record). This forces new data files to be created each time the 255 record boundary is filled. Also, any formatted file read by the 8600 CPU cannot be larger than 255 records. The record size constraint creates further overhead in uploading data files from the AB 8600 to company business systems. NC part program files are unformatted so they can be as large as memory allows. Deletion of files requires a manual key insertion and editing privileges. Thus, operator lockout was not possible, so data integrity could not be assured. ie; operator can modify production files.

Software maintenance is difficult and costly due to the structure of SIPROM code and the size of the PAL code running on the PLC's. The single biggest problem with this system is lack of memory. The machine controller is running at maximum capacity. Because additional memory is unavailable, no new process can be added to this system. For example, adding another RS-232 port would require memory to set up a serial communications structure. Any modifications to existing code is very difficult. On the other hand, this system is currently exceeding production goals in the factory.

System 4

The retrofit system consists of a 5 axis JOBS Jomach 16 with a sixth W feed axis, and a spindle. This system is interfaced to an Adept A-series IC controller. The purpose of the retrofit system is to provide a test and feasibility workcell for various automation processes under development within Boeing. Figure 4 depicts the hardware block diagram for this system. The system goal was to be extremely flexible, accommodating diverse applications.

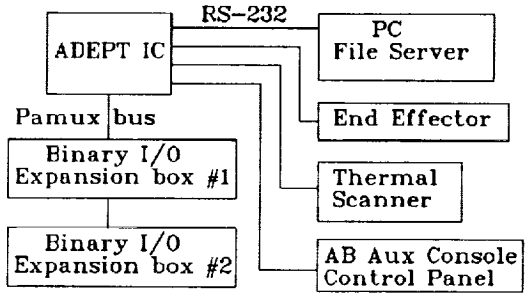

Figure 4

Number of axes	7
Number of I/O's	100+
Number of serial ports	1
End Effector	Multiple
Languages	V/V+
Lines of code	15,000+

Table 4

The system uses 4 serial communications ports. One is connected to an external PC for file transfer. Another is connected to an operator control console (OCC). The third is connected to a thermal scanner, and the last is used for communicating to an end effector control system.

The system uses more than 100 digital I/O's for process control. Most of these are used in control of spindle operations. Digital I/O is split into three groups: input, output, and interrupt functions. Each of these groups can be subgrouped into banks of 8 discrete I/O's for partitioning in software. The interface to the OCC uses both RS-232, interrupt, and digital inputs. Because cycle start and cycle stop functions are critical to NC operations, a non-maskable interrupt is used to acknowledge input form the OCC.

The Adept controller provided many functions used in serial communications. For example, file transfer functions from the PC to the Adept are buffered. Although an in house transfer protocol is used, Kermit or Xmodem could have been applied. Because the amount of data read from the thermal scanner is small compared to file transfer, communication is done asynchronously without buffering.

The retrofit project benefits from using one language capable of controlling I/O's, interfacing to an operator console, defining serial communication formats, and developing decision paths for the application software. The language is efficient in supporting variable definition. For example, a program must perform automatic range changing of the spindle drive gearbox. The application code was written using variables such as **sp.in.rng.1**, and **sp.in.gear1.i** to define the spindle gear range and state of the gear 1 input sensor.

The tools for graphics were used extensively in developing user interface screens. Features such as buttons, icons, window and scrolling were implemented in most of the application software. The language also supported structured techniques which allowed for modularizing the application code. Because of this, many code modules are being reused in other applications currently under development. On the other hand, the language V+ is proprietary to the controller and required some training before programming could begin. The controller fully supported RS-232 and file transfer functionality but was not equipped with protocols such as Ethernet, SNA, or MAP. This shortcoming provided difficulty in interfacing to company business systems.

Maintenance and life cycle costs of the software are difficult to determine because code is always being developed for new applications. It should be noted that by developing modular functions and meaningful variable definitions, most of the application code is understood by reading it directly. Electrical maintenance of the system is undetermined because the machine has not broken down yet. Mechanical functions remained the same after the integration.

NC Translator Application

In addition to the four previous case studies, there was a requirement to develop an NC translator which could read NC code developed for system 3 and execute it on system 4. The application required exact replication of NC motion with a control system using kinematic trajectory generation. The Adept controller uses built-in kinematics during trajectory calculations. The kinematic definition of the machine includes link lengths, joint angles, joint configurations etc. The NC translator application required encoding the NC joint positions into WORLD coordinates for use by the control system's trajectory generator. An NC controller moves the machine joints to locations using linear or circular interpolation. The G-codes being executed by the NC machine determines the type of interpolation employed. Conversely, a robotic controller uses kinematics to compute trajectory points for driving the tool tip. The robotic controller can then use linear or joint interpolation to drive the machine in WORLD, TOOL or JOINT space.

Path motion created unique problems with respect to accuracy. A path may be represented by a series of consecutive points. As the tool tip moves through these points several events occur. The tool tip moves toward point 1 while the control system is computing a trajectory for point 2. As the tool tip approaches the target point 1, it may move through that point or come close to it as it moves towards point 2 in the path. The controller looks ahead 1 point in the path and computes a trajectory to that point. At some time in the trajectory, the tool tip begins to move towards point 3 and so on. The velocity and acceleration values directly affect the accuracy of the tool tip in following the prescribed path. In machine routing, the smoothness of the motion over a path is critical to the quality of the new surface left behind by the router blade. A constant velocity is required to make a smooth cut.

The controller allows for tuning envelopes around endpoints in motion but did not allow for definition of a tolerance envelope around path points. A solution required close spacing of path points in the NC program. During path motion execution, the next point in the path was broken down into a series of smaller constant velocity moves. The machine structure of 5 axes together with path slicing computations produced two wrist configurations for the same point. Additional software was written to assure wrist configuration was maintained during path motion. The result allowed the machine to follow paths dictated by RS274D G-codes even though the trajectories were computed using forward and inverse kinematics.

The NC translator requirements included simultaneous execution of the following functions: a graphics display including which NC block was currently executing; real time monitoring of an auxiliary operator control console; preparation of path points for tool trajectory; executing proper motion as defined by RS274D G-code standards.

The application required the use of 3 tasks and several internal software flags for inter task communication.

The multitasking capability of the operating system was invaluable in coordinating the 3 application tasks. Software was used to set task prioritization and optimize stack sizes. This application is currently used to test NC programs for developmental assembly concepts.

Future Work

If robots and machine tools are to realize their full potential, controllers must improve their computational performance, support reusable software and provide for system extensibility. Open architecture controllers based on accepted industry standard hardware, operating systems, and application languages are arguably the best way to support these improvements.

Machine controllers are typically two generations behind the best available microprocessors. This performance lag occurs due to lack of portability of control software as well as robot and machine tool suppliers using proprietary high level and assembly programs to implement unique mini-kernels in lieu of a conventional operating system. Control software written in ANSI C with careful conformance to POSIX standard system calls can be ported to new processors in a matter of days. The use of standards further encourages software re-use, since application code can often be re-compiled in the new environment and linked into higher level software designs.

Robot system extensibility demands a computing hardware environment that enjoys high volume use and a spirited development community to ensure an uninterrupted stream of hardware to support emerging requirements.

Boeing, in support of this approach, is developing open architecture controllers and motion control libraries in cooperation with several commercial vendors. The robot controllers are VME based, programmed in ANSI C and are POSIX compatible. Extensions to this work will provide retrofit software applications to ease the adaptation of open controls to new machines. Servo tuning tools, simulation systems, calibration applications, and upstream system interface libraries will be developed during the next year or two.

The author would like to thank the following dedicated automation and robotics engineers at Boeing for their experienced input: Craig Battles, Rich Morihara, Stan Munk, and Scott Muske.

References

[1] Reintjes, J. Frances. 1991 *Numerical Control: Making a new Technology*, New York. Oxford University Press.

C-2