NISTIR 6753

Porting a Complex Machine Tool Error Compensation System From Microsoft DOS to Microsoft Windows NT

Herbert T. Bandy
M. Alkan Donmez
Lawrence A. Welsch

1901-2001
NIST CENTENNIAL

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

NISTIR 6753

Porting a Complex Machine Tool Error Compensation System From Microsoft DOS to Microsoft Windows NT

Herbert T. Bandy
M. Alkan Donmez
Lawrence A. Welsch
Manufacturing Metrology Division
Manufacturing Engineering Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8220

July 2001

Abstract

This report describes the porting of a manufacturing quality control system from PC-type computers operating on Microsoft DOS, Microsoft Windows 95, and IBM OS/2, to updated PCs operating on Microsoft Windows NT. The changes included adapting the control software and input/output (I/O) card drivers from 16-bit requirements to 32-bit requirements. Real-time response was a critical issue. The programming environment was also changed to better integrate the modules and languages.

1. Introduction

The system discussed is designed to implement closed-loop error compensation on a Monarch Metalist turning center.[1] It was developed to improve the quality of part fabrication using the strategies of Quality in Automation (QIA). QIA is a system of quality control/quality assurance strategies developed at the National Institute of Standards and Technology (NIST) using an architecture with multiple feedback loops to control the process for small-batch manufacturing [Donmez, 1991]. The system software is hosted on three computers. One, called the Quality Monitor, is used as a data repository and analysis host. Another, called CMM, controls a coordinate measuring machine. The third computer, the Quality Controller, provides custom control of certain machine tool functions. Data management and networking integrate the software across all hosts.

The system performs process-intermittent and post-process inspection of parts for dimension, form, and surface roughness. It compensates machine geometric-thermal errors and process errors, and refines the error compensation algorithms according to post-process residual errors [Vorburger, 1994].

The system was recently ported from PC-type computers operating on Microsoft DOS, Microsoft Windows 95, and IBM OS/2, to updated PCs operating on Microsoft Windows NT. On the old hosts, the system had proven to work adequately within limits, but needed improvement in the performance of its functions, its capabilities expanded, and consolidation of languages and modules. An upgrade in the computers necessitated new drivers for hardware, and many software changes for application compatibility. The Quality Controller presented the main challenges, since it interfaces with several devices. The old Quality Controller had a 60 MHz Intel 80486 processor, 16 MB of random access memory (RAM), and operated on Microsoft DOS 5.0. The new Quality Controller has dual 200 MHz Pentium Pro processors, 128 MB of RAM, and operates on Microsoft Windows NT 4.0.

[1] Certain equipment, instruments, or materials are identified in this report in order to adequately specify certain experimental procedures. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the material or equipment identified is necessarily the best available for the purpose.

The QIA system applies specifically to machine tools, and is implemented on a turning center in the prototype discussed here. To establish the context for further discussion, machine tool components and errors will be explained next.

1.1 Machine Tools

Metal cutting machine tools are used to produce parts by removing material from a part "blank," a block of raw material, according to the final desired shape of the part. In general, machine tools consist of components that hold the workpiece and cutting tool. By providing relative motion between these components a machine tool generates a cutting tool path which in turn generates the desired shape of the workpiece out of a part blank. The metal cutting energy is generated by either rotating the workpiece (for turning processes) or by rotating the cutting tool (for milling processes). The nominal shape and the tolerances on the dimensions, shape and surface finish of the desired part determine the type of machine tool and cutting process to be used.

In modern machine tools, the relative motion between the cutting tool and the workpiece is controlled by a computer, which is called the Computer Numerical Control (CNC). The CNC unit of the machine tool generates the instantaneous target positions to be reached by each slide (axis) of the machine based on the input numerical control (NC) part program, which describes the desired part geometry. It coordinates the motions of multiple slides with respect to each other to make sure that the resultant geometry is accurate. In addition, the CNC carries out the auxiliary functions such as turning on and off the coolant system to remove heat generated by the cutting action, locking and unlocking of the part on the machine table, etc.

Machine tools operate using multiple slides, rotary tables and joints, which are usually mounted on top of each other, each moving along a single direction of motion. The number of slides that are independently controlled and coordinated with the other slides determines the number of axes of the machine tool. Each slide or rotary table of a machine tool is designed to have a single degree of freedom (translation or rotation). In other words, each slide is designed to move in only one direction. However, due to imperfections in design and manufacture of machine tool components and their assembly, each slide in a machine tool exhibits error behavior with six components. With additional angular errors of one slide relative to the other, a 3-axis machine tool has 21 error components [Soons, 1992]. The result of all these errors is the deviation of part dimensions and form from their designed/desired specifications. Therefore, to produce high quality parts, one has to minimize these errors of machine tools.

Since most of the errors mentioned above are repeatable and predictable, one approach to minimize them is to develop predictive models of such errors and store them in computers to compensate for these errors in real time. The QIA architecture developed at NIST helps realize this strategy using a PC-based quality controller that is interfaced with the machine tool CNC to improve its performance during machining. The machine tool, a turning center, used in this study is shown in Figure 1. A typical part that is produced on such a machine is shown in Figure 2. The next section will describe the error compensation system implemented on the turning center.

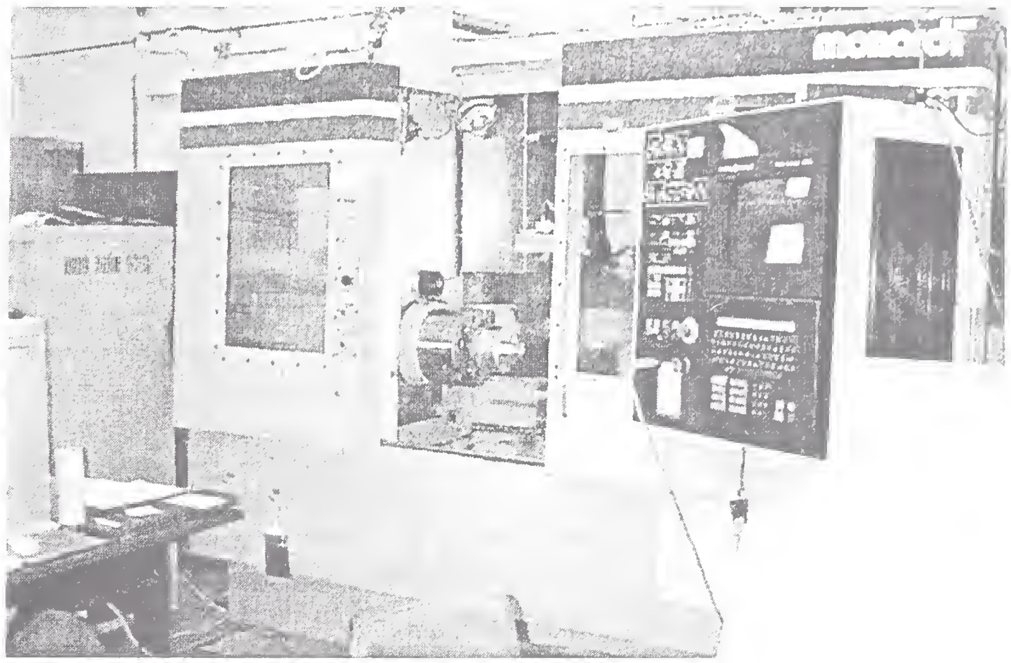


Figure 1: Turning center used in the study

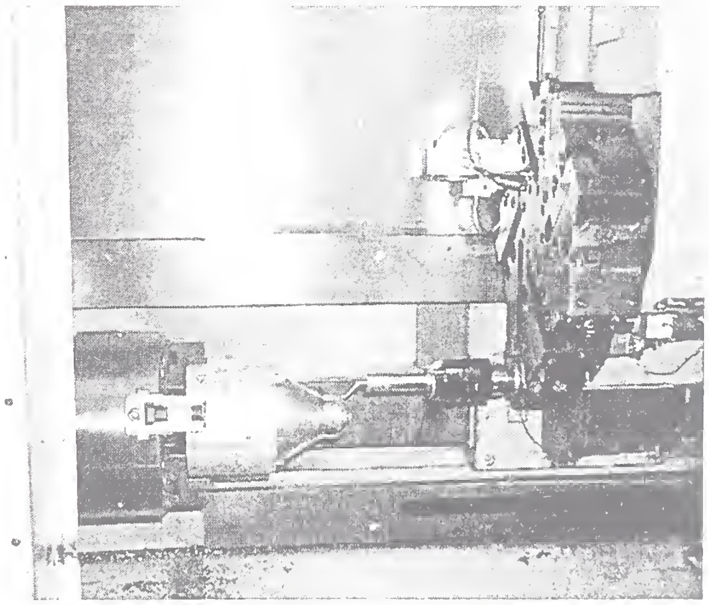


Figure 2: A typical part being inspected on the turning center

2. Description of the Error Compensation System

The basic control strategies [Donmez, 1992], data management [Bandy, 1995b], and prototype software functions [Bandy, 1995a], have been documented elsewhere. This section explains the relationships between the following major components of the architecture.

Machine Tool:	Monarch Metalist Turning Center.
CNC:	GE-2000T.
Quality Monitor:	PC-type computer hosting the database and programs for data analysis and algorithm refinement.
Quality Controller:	PC-type computer hosting the QIA control program, the geometric-thermal model of the machine tool, and other process-control programs.
RTEC:	The Real-Time Error Corrector is a "black box" which serves as an interface between the Quality Controller and certain CNC functions [Yee, 1992]. The RTEC facilitates the error compensation activities conducted by the Quality Controller. Most significantly, the Quality Controller may control the tool path by causing the RTEC to modify the position feedback signal from the machine axes before it is passed on to the CNC.
Probe Controller:	For on-machine dimensional gauging.
Surface Sensor Controller:	Includes a signal processor and rotary positioning device for on-machine surface inspection using an ultrasonic sensor.
CAD-Based Functions:	Functions using a Computer-Aided Design system are the basis for the part design, feature definition, specification of inspection paths, development of NC part programs for on-machine inspection, creation of the Dimensional Measuring Interface Specification (DMIS) part description [ANSI/CAM-I, 1989]. This vital group of strongly interrelated functions is distributed between the three PC-type computers.
CMM/CMM computer:	Giddings & Lewis Cordax 5 coordinate measuring machine (CMM) controlled by a PC-type computer.

The basic interactions between most of the above architectural components are indicated in Figure 3. The interactions will not be elaborated here; the requirements for communication are more pertinent to the topic of this report.

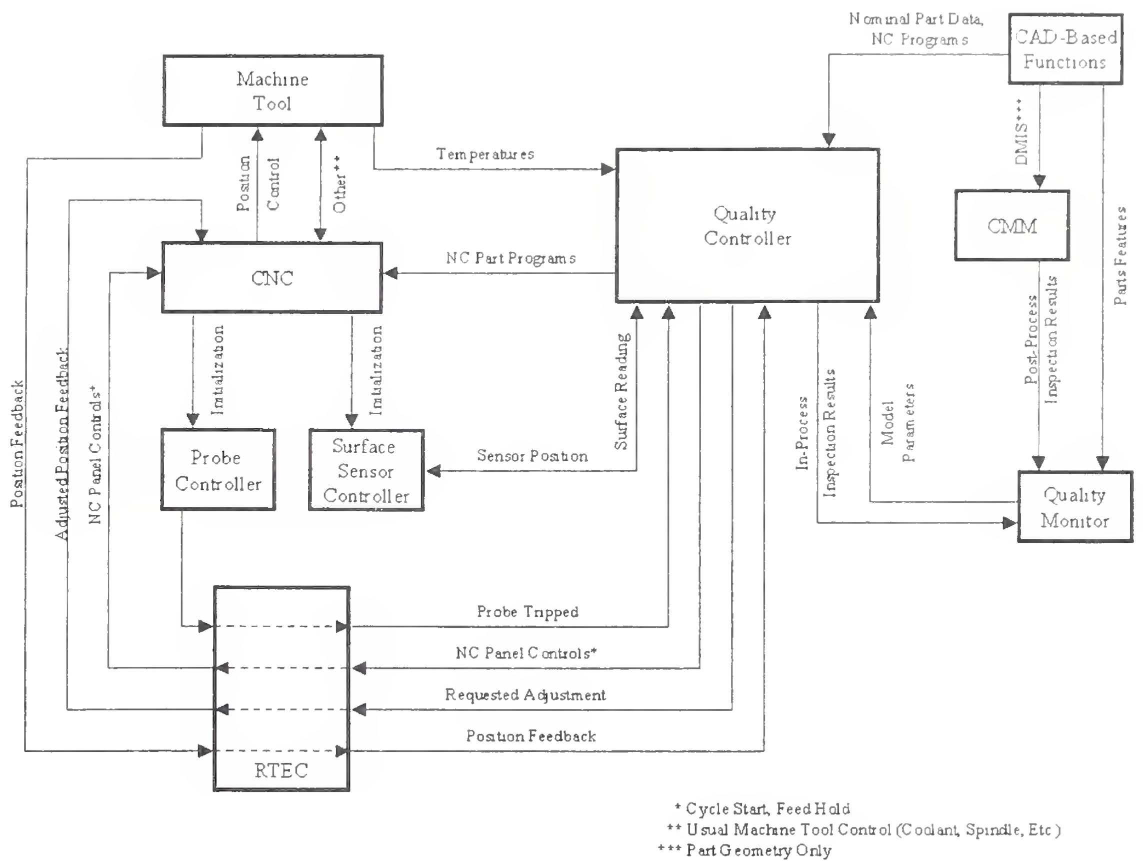


Figure 3: The major components of the QIA architecture

A data acquisition control unit reads machine tool temperatures with thermocouples, and passes the data to the Quality Controller through an IEEE-488 interface card. NC part programs are transferred to and from the CNC by custom software using serial communications. The rotary indexer that positions the surface sensor uses serial communications with commercial software. Data from the ultrasonic surface sensor itself is read through an analog-to-digital (A/D) card. Data communications between the three PC computers (the Quality Controller, CMM, and the Quality Monitor) is accomplished with Microsoft networking. The RTEC, a unique electronic interface, was developed to mediate most other communications. The control software on the Quality Controller sends data to and reads data from the RTEC through a digital I/O card. In turn, the RTEC reads probe-trip signals from the probe controller and reads axis positions from the machine tool. It also sends panel control signals and machine position adjustments to the CNC.

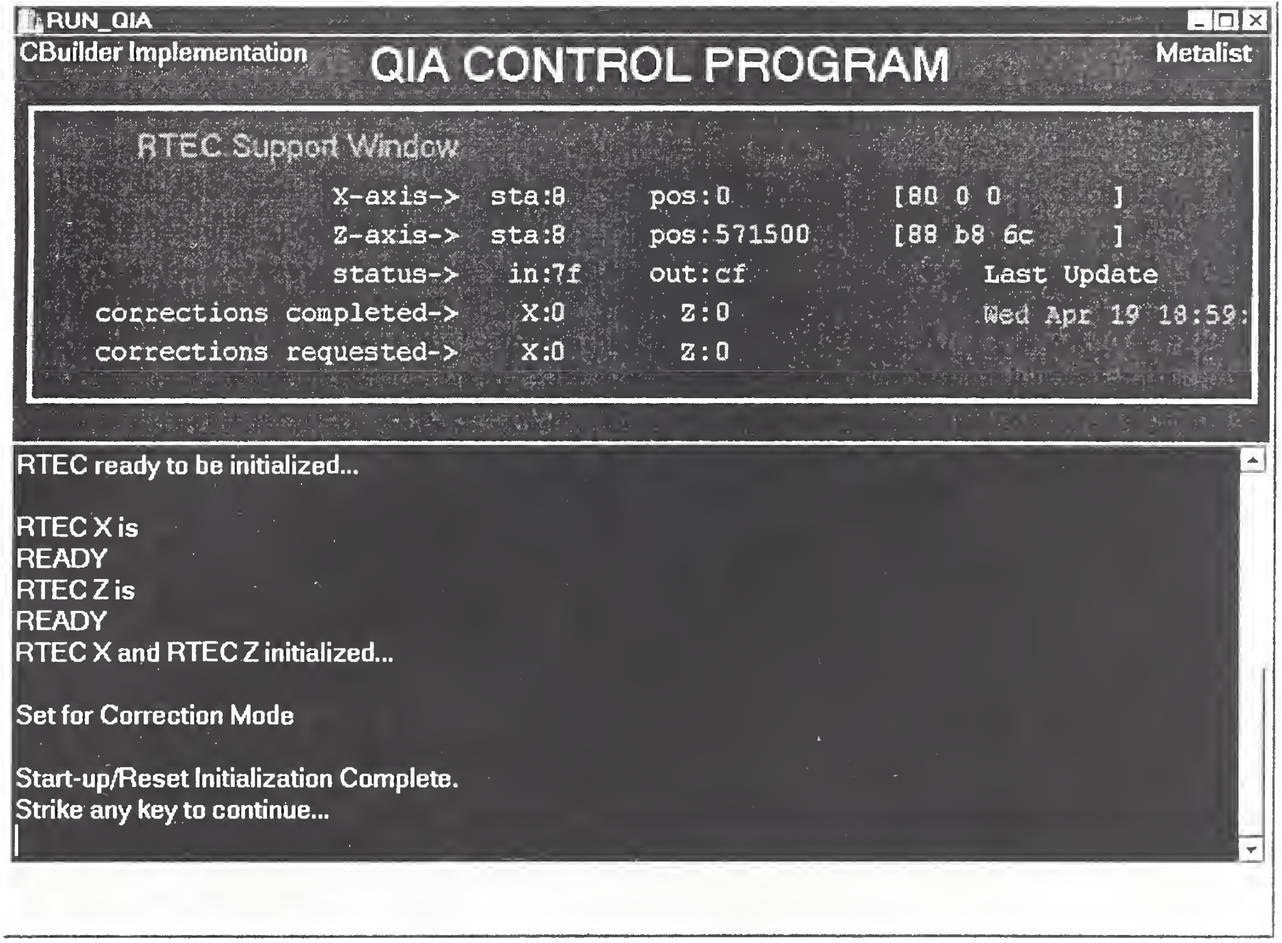
3. Porting to New Hosts

The QIA system software consists of about 50,000 lines of code in 165 files. All software functions support the improvement of part quality on the turning center. The QIA Control

Program, the major module hosted on the Quality Controller, executes the tasks involving direct interaction with the turning center. Those tasks are 1) transferring NC part programs to the CNC; 2) starting execution of NC part programs; 3) controlling real-time error compensation during machining; 4) collecting dimensional data on part features; 5) collecting temperature data; 6) storing the data in a database; and 7) analyzing the data. The original QIA system ran on DOS 5.0, and the undertaking reported here was to make it run on Windows NT 4.0. The challenges of porting the system included 1) changing the control display to a Windows graphical user interface (GUI); 2) porting the RTEC and the digital I/O interface used to drive it; 3) porting device drivers on an IEEE-488 bus; and 4) porting the software that controls the serial I/O ports.

3.1 User Interface Challenges

Figure 4 shows the GUI for the QIA Control Program after it was modified to run on Windows NT. The interface has essentially the same appearance as it did when running on DOS. The differences are 1) the DOS program display did not have a title bar across the top; and 2) the lower panel of the DOS program display did not have a scroll bar on the right. The similarity in appearance was desirable to facilitate user understanding of the updated software.



```
                  X-axis->  sta:8     pos:0          [80 0 0      ]
                  Z-axis->  sta:8     pos:571500     [88 b8 6c    ]
                  status->   in:7f    out:cf              Last Update
   corrections completed->    X:0       Z:0               Wed Apr 19 18:59:
   corrections requested->    X:0       Z:0
```

```
RTEC ready to be initialized...

RTEC X is
READY
RTEC Z is
READY
RTEC X and RTEC Z initialized...

Set for Correction Mode

Start-up/Reset Initialization Complete.
Strike any key to continue...
```

Figure 4: The graphical user interface for the QIA Control Program

The user interface is composed of two distinct parts. The upper window displays state data reported by the RTEC, such as the position of the machine axes and tool-position corrections completed for each axis. The state changes as the NC part program is executed. The bottom window shows system status and prompts to the user. One improvement implemented when moving to the GUI was that a scroll bar was added so that statements that scrolled off the top of the screen could be seen.

The traditional DOS-based user interface required a series of blocks of code, the execution of each depending on specific user inputs being detected by system polling. In the new user interface design, program responses are executed when interrupts are generated by GUI events, as illustrated in Figure 5. We chose to implement the new GUI in a separate thread from the thread of the main program. We modified the interface procedures in the main thread to use the GUI thread to either read or write data. We synchronized the threads when reading or writing. While the flow of control in the main program remained the same as before, the event-driven operation consolidated the interface.

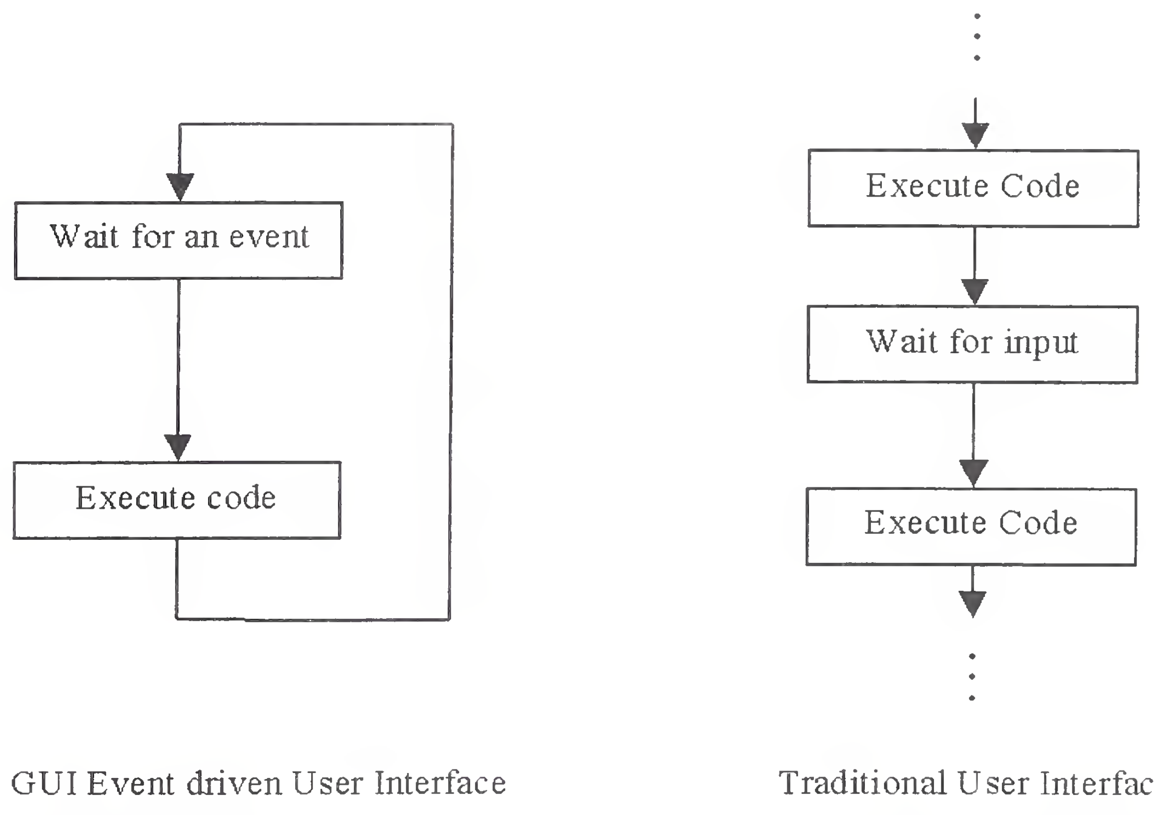


Figure 5: The new GUI event handler executes code that formerly depended on various specific user inputs

3.2 Digital I/O Interface Challenges

At the start of the porting effort, we were seriously concerned about having to write a special device driver for use with a new Windows NT-compatible digital I/O card to interface to the RTEC. After extensive searching, we located a device driver that allowed the NT-based QIA system to communicate at the application level with the same digital I/O card used with the DOS-based system. We devised procedures with test inputs to carefully check card output and RTEC response. Before putting the card in operation with the system, we had to ensure that it was properly reading and writing the same data when used through this interface as it did when we used the DOS interface. For all the concern we had, there were no problems.

3.3 IEEE-488 Bus Challenges

For the Quality Controller to read machine tool temperatures from the data acquisition control unit, we bought a new Hewlett-Packard 82350 PCI HP-IB interface card for Windows NT computers. For new applications and for VXI*plug&play* compliance, Hewlett-Packard recommends using the Virtual Instrument Software Architecture (VISA) I/O library for use with the HP-IB interface. However, the company also provides an alternative for compatibility with older software, the Standard Instrument Control Library (SICL) [HP, 1996]. Using the SICL I/O library, the porting problems were minor. SICL commands are different from those used with our old DOS software, but similar. We had to manually locate the differences and rewrite several procedure calls. The job was further complicated by the fact that the documentation for the original HP-IB DOS interface software was unavailable.

3.4 Serial I/O Challenges

We had the most difficulty with the serial I/O interfaces. We originally implemented them in DOS using BIOS procedure calls. This gave a great deal of control to the application program. However, Windows NT receives and sends data on the serial I/O devices by using generic READ and WRITE calls, making programming simpler, but permitting less direct control over communication parameters. This is the one place in the program where we had to rewrite the entire interface and the calls. We also spent more time debugging and testing the serial I/O interface than any other part of the system. Today, although we are able to accomplish serial communications, it is still the part of the system we feel needs reliability improvements.

3.5 Future Enhancements

One of the objectives of upgrading the system was to consolidate languages and modules. To some degree this was done. The introduction of Borland C++ Builder 4 as the programming environment for much of the code replaced the scattered use of Microsoft C, Borland C++, and some of the code written in the Cadkey Advanced Design Language.

But further enhancements are still desired. There is still some dependence on application-specific database functions that we would like to supersede. Open Database Connectivity (ODBC)

functions in the Builder environment could replace the use of the Paradox Application Language and the Paradox Database Engine. To facilitate future development, the early C code should be restructured to be object-oriented. And further enhancements to the serial communications software subsystem would improve reliability.

The A/D interface card formerly used on the old Quality Controller to collect data from the ultrasonic surface sensor was not installed on the new computer. This does not matter at the moment because the surface inspection apparatus has been removed from the machine tool for offline development. But for the Quality Controller to be ready to perform all its functions, the PCL-812PG A/D card should be installed and tested.

4. Conclusions

There are several clear benefits of having a common, upgraded platform for the three main computers of the system. Unlike before, CAD-based part geometry data from the other two computers can now be shared with the CMM computer, enabling code for running the coordinate measuring machine to be more easily produced. Also, because of the increased speed and memory, more interrelated programs can run together. Before the upgrade, the analysis programs of the Quality Monitor had to run in sequence.

More of the benefits are obvious for the Quality Controller. The system is much more stable than before, never locking up from memory management problems. Quality control operations can now occur smoothly, rather than each software function needing to exit before another can start. Modules performing data collection, CAD plots of part errors, calculations of error compensation parameters, and machine control can now run as needed. The increased memory also allows a part to be inspected at more points than before, since a significant amount of data is carried with each point.

The computer speed also helped the real-time response of error compensation. On the old host, the rate of tool path adjustments was limited by the speed at which they were calculated. Now the calculations are executed well within the preferred 20 ms cycle time for updating the tool path. Another benefit of speed is that the data window is updated at a sufficient rate to avoid screen flicker.

In addition to system performance improvements, it is also easier to maintain and enhance the software now that more modules and languages have been consolidated and integrated in one programming environment.

5. References

ANSI/CAM-I. *Dimensional Measuring Interface Specification, Version 2.1, R-89-DMIS-01,* Computer Aided Manufacturing - International, Arlington TX; 1989 July.

Bandy, H.T.; and Gilsinn, D.E. "Compensation of Errors Detected by Process-Intermittent Gauging," *ASPE 1995 Annual Meeting,* Proc. American Society for Precision Engineering, 12:440-443; 1995 October.

Bandy, H.T.; and Gilsinn, D.E. "Data Management for Error Compensation and Process Control," *Modeling, Simulation, and Control Technologies for Manufacturing,* Proc. SPIE, 2596:114-123; 1995 October.

Donmez, M.A. "Development of a New Quality Control Strategy for Automated Manufacturing," *Manufacturing International—1992,* ASME; 1992.

Donmez, M.A. (ed.) *Progress Report of the Quality in Automation Project for FY90,* NISTIR 4536, National Institute of Standards and Technology (U.S.); 1991 March.

HP I/O Libraries Installation and Configuration Guide for Windows, Hewlett-Packard Company, third edition; 1996 October.

Soons, J.A.; Theuws, F.C.; and Schellekens, P.H. *Modeling the Errors of Multi-Axis Machines: A General Methodology,* Precision Engineering, 14(1): 5-19; 1992 January.

Vorburger, T.V.; Yee, K.W.; Scace, B.R.; and Rudder, F.F. Jr. "Post-Process Control of Machine Tools," *Manufacturing Review,* 7(3): 252-266; 1994 September.

Yee, K.W. *Alternative Designs of a Real-Time Error Corrector for Machine Tools with "Encoder" Position Feedback,* NISTIR 4832, National Institute of Standards and Technology (U.S.); 1992 April.