Design and Implementation of a Fault Tolerant Magnetic Bearings Controller

Stephen J. Fedigan[1]
Ronald D. Williams[1]
Feng Shen[1]
Robert A. Ross[2]
[1]Center for Magnetic Bearings, University of Virginia, Charlottesville, VA, U.S.A. 22903
[2]Center for Semicustom Integrated Systems, University of Virginia, Charlottesville, VA, U.S.A. 22903

Abstract

A digital controller designed to meet the demanding performance and reliability requirements of high-speed magnetic bearing applications is presented. The controller architecture is a fault-tolerant design, consisting of four processor modules arranged in a triple modular redundant (TMR) configuration with a hot spare. A voting module provides active repair and reconfiguration in the event of data disagreements or "lost" processors. The processor modules are independent computers based on the Texas Instruments' TMS320C40, a high speed 32-bit floating point digital signal processor (DSP), capable of executing 220 million operations per second (MOPs). Reliability is further enhanced by a fault tolerant I/O bus, whose data / address buses are each augmented with a set of check bits for single-bit error correction (SEC). Support for the development of control algorithms is provided by a multi-tasking, real-time operating system (RTOS).

Introduction

As magnetic bearings find greater application in the support of high-speed rotating machinery, digital control systems will be required to execute complex control algorithms with high sampling rates. In addition to providing significant digital signal processing power, these platforms must be highly reliable over long time periods to be cost effective and practical in industrial settings, or to be used in safety critical applications. A fault-tolerant digital controller recently completed at the University of Virginia's Center for Magnetic Bearings delivers both the high performance and high reliability needed for such applications.

Background

The Center for Magnetic Bearings at the University of Virginia has been conducting magnetic bearing research in a wide range of areas since 1984. This research has included work in analog controllers, digital controllers and algorithms, power amplifiers, industrial applications, and magnetic bearing/rotor system dynamics [1].

Several digital controllers, tailored for magnetic bearing control, have been designed and built over the past ten years. The first effort consisted of an IBM-type personal computer, using a multiplexed analog I/O subsystem. This simple system successfully supported a small, flexible rotor, but the controller sampling rate was restricted by the 8086 CPU to approximately 3 kHz [2].

The second controller improved performance by creating an architecture customized for magnetic bearing control. This design employed many separate digital control modules (DCM) for executing single-axis control, governed by a supervisory computer. Each DCM featured a Texas Instruments TMS320C25, a fixed-point DSP [3]. This work initiated the use of DSPs for magnetic bearing control at CMB, a practice that continues in the present design effort.

The third-generation digital controller, the Integrated Controller for Magnetic Bearings (ICMB), was designed using a single CPU, the TMS320C30 floating-point DSP, with modular analog I/O capability [4]. Additional system hardware, such as a Motorola 68HC11 microcontroller, supports the 'C30, so that its effort may be focused on executing the control algorithm. A real-time, multi-tasking operating system, has been developed for this system, easing algorithm design, development, and evaluation [5]. As a result of this work, the ICMB is in near constant use by a variety of CMB projects for experimentation with and development of magnetic bearing control algorithms.

System Requirements

Although high reliability was considered the primary objective in the controller design, methods which sacrificed performance were deemed unacceptable. This ruled out time and software-based information redundancy techniques [6], which impose processor overhead and reduce the achievable

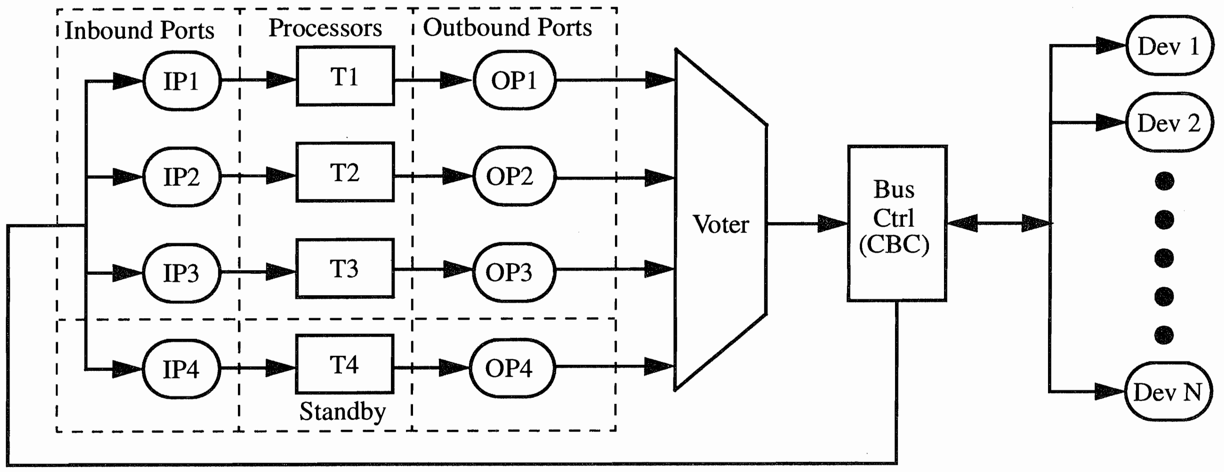


FIGURE 1: System Architecture of Magnetic Bearings Controller

sampling rates. Hardware intensive methods, such as hardware redundancy or hardware-based information redundancy, were preferred in developing an architecture suitable for the magnetic bearings application. Such techniques have the added advantage of being transparent to the application designer.

Besides fault tolerant techniques, fault avoidance methods were employed to boost the controller's reliability. This was accomplished through the use of surface mount vs. through-hole components, which offer better noise immunity, require fewer solder joints, and can tolerate greater exposure to shock and vibration [7]. The extensive use of programmable logic devices (PLDs) reduced the component count and hence the number of solder connections on the board.

Over the past four years, a suite of control applications have been developed on the University of Virginia's 'C30 based hardware platform, including routines that implement high speed digital filters and adaptive on-line balancing [8]. For the most part, these routines have been written in C, with certain time-critical portions written in 'C30 assembly language. The code has been compiled with TI's TMS320 Optimizing C Compiler. Additionally, these routines include service calls to a real-time operating system. Since considerable effort has gone into writing and debugging existing application programs, an important requirement is portability between the third and fourth generation digital controllers. Compatibility is ensured by using the TMS320C40, whose instructions are a superset of the 'C30's, by using the same suite of development tools, which can generate code for either the 'C30 or the 'C40, and by retaining the same application program interface (API) in the advanced version of the RTOS.

System Architecture

To satisfy the need for high performance and high reliability, the fourth generation digital controller realizes fault tolerance through hardware redundancy, and hardware-based information redundancy. As shown in the Figure 1, the controller's system architecture consists of four digital signal processing modules, arranged to provide triple modular redundancy with a hot spare. Each processing module is an independent computer, whose core is a TMS320C40 with its own local oscillator and local memory bank. Each processor's control actions are sent out of a communications port to the voter subsystem, which provides active repair and re-configuration in the event of a processor failure. The voter essentially acts as a sophisticated multiplexer, passing the results from the processor whose data is determined to be correct to the channel bus controller (CBC). This next subsystem performs the requested transfer between the processors and a specified I/O device across a fault tolerant, 4.0 MHz synchronous bus. These I/O devices include 8 channel 14-bit A/D cards, 8 channel 14-bit D/A cards, digital I/O cards, and RS-232 serial ports which support interaction between the controller and a host computer.

Processing Modules

The computing power of the fourth generation digital controller resides in four independent, identical computer modules. Each one of these is a separate printed-circuit (PC) board, which plugs directly into the motherboard. With this configuration, modules can be replaced easily if repairs become necessary, and maintenance and development of application software is greatly simplified.

Each module is centered around a TMS320C40 DSP operating at 40.0 MHz, with a 64k x 32 bank of zero wait-state SRAM, and a 128k x 8 bank of EPROM. Besides the compatibility issue, the 'C40 has been chosen because it offers a mature set of development tools (an optimizing C compiler and a source level debugger) and has features that make it attractive for control applications. Like most DSPs, it offers a single-cycle floating point multiply and accumulate, and a circular addressing mode, making it ideal for digital filtering applications and Discrete Fourier Transforms (DFTs). Many common DSP / mathematical operations enjoy direct

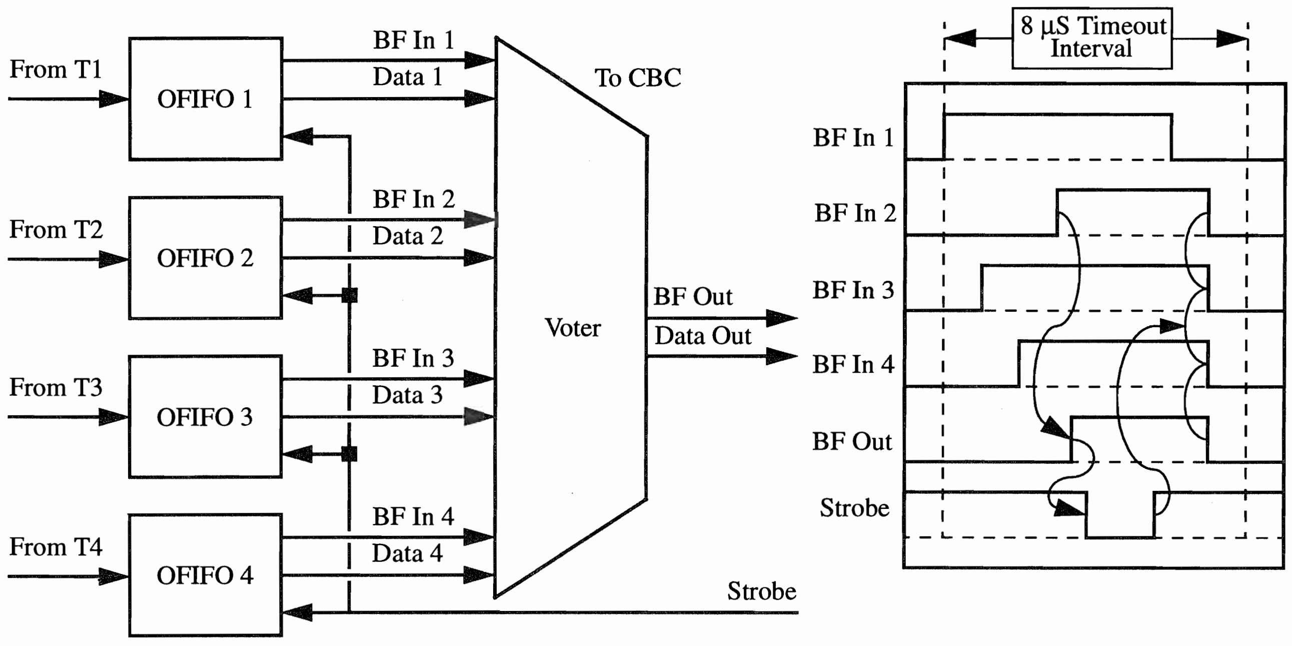


FIGURE 2: (a) Voter unit with output FIFOs and (b) Output synchronization in the normal operating mode.

hardware support. For example, a bit-reversed addressing mode enables the DSP to quickly order the results of a Fast Fourier Transform (FFT), and single instructions are available for calculating a floating point reciprocal and an inverse square root. The DSP has a high performance architecture, which includes a four stage pipeline, a bank of single-cycle dual-access memory, and a sophisticated six-channel direct memory access (DMA) controller, which can operate in a chained-transfer mode. This means it can perform a memory transfer and then re-initialize itself and start a second memory transfer. In other words, a whole sequence of memory transfers can be executed without the need for any CPU intervention.

For multi-processing applications, the ‘C40 is equipped with six parallel, bi-directional communications ports, each capable of transferring data at 20 MW/s. In this system, a dedicated inbound port receives sensor data from input devices on the bus, and a dedicated outbound port sends control actions to the voter.

VOTER

In normal operating mode, all four modules execute the same control software and generate a series of control actions every sampling period. These actions are translated into bus operations, and are sent through each outbound communications port. Each word in this data stream passes onto the voter shown schematically in Figure 2(a), which performs a bitwise majority vote on the results arriving from processors T1, T2, and T3. To make sure that all processors move in lockstep despite variations in their local oscillators, data arriving from each communications channel is stored in an outbound FIFO, until results from all four processors become available, as indicated by a buffer full signal. As shown by the timing diagram in Figure 2(b), once all of these signals have been asserted, the voter latches the results of the majority vote, and removes the leading word from the head of all FIFOs. This is done even for the hot standby, to guarantee that it remains synchronized with the voting processors. In addition, synchronization is maintained at the beginning of each sampling period when the sensor data becomes available. The voter waits until each one of the inbound FIFOs asserts its buffer empty signal, and then loads all four in parallel. This assures that the sensor inputs arrive at each processor at the same time. Thus, the voter provides both fault masking through data voting, and data synchronization by updating the input and output FIFOs at the same time.

If T1, T2, or T3 fails during normal operation, its failure is detected if it sends out two consecutive words of bad data, or if its FIFO control signals disagree with the others for more than 8 μS. If T4 fails in one of these two ways, the event is noted but no action is taken until another processor fails and T4 becomes one of the voting processors. When a failure is indicated by T1, T2, or T3, the voter removes the offending processor, which no longer participates in data voting, and replaces it with the standby spare. In re-configured mode, the standby and the two remaining processors vote on the data coming from the outbound FIFOs. Bad data from one of the three does not cause further re-configuration, because it is masked by the other two. However, if the control signals from one of the processors disagree for more than 8 μS, the voter switches to simplex mode, selecting the working processor with the lowest number.

One of the important goals in the design of this voter is to minimize the chance of a false alarm. For example, a single isolated word of bad data from a voting processor during normal operation does not trigger a re-configuration. The voter distinguishes between transient errors which might be caused by intermittent noise and are masked out, and static errors which are a stronger indication of a processor failure. The control signal watchdog window has been very conservatively chosen so that the chance of a time-out under normal operation is minuscule. A 8 μS latency in 50 μS sampling period is a clear indication that the processor has gotten "lost".

Other voter design goals include minimizing the effect of a re-configuration on the control system, and minimizing the voter's effect on system performance. In reference to the former goal, re-configurations caused by bad data never cause any added delays, or produce bad data. However, re-configurations due to "lost" processors do cause an 8 μS delay in the delivery of a series of control actions, which should be a small enough percentage of the overall sampling period to avoid serious problems. In reference to the later goal, the frequent re-synchronization of the processors prevents large time skews in the arrival of data from the processors. Furthermore, the data voting is realized with purely combinational circuits and has been designed such that it adds only a small transport delay to data moving from the processors to devices on the bus, but does not affect the bus transfer rate.

Since the voter represents a single point of failure, it must be highly reliable. This issue has been addressed both in both its design and implementation. From the design side, the voter's state machine is synchronous and only has three states which represent the three operating modes: normal, re-configured, and failed. With a small number of *states*, the design can be verified easily in simulation and in the lab, and the state machine can be designed by hand, without resorting automated synthesis tools, which can introduce logic hazards. Since the voter runs on the same clock as the bus controller, control lines which pass between them do not need to synchronized and metastability problems are avoided completely. From the implementation end, the voter's logic is embedded in three PLDs with good reliability records.

Bus Controller

Data from the voter is sent on to the channel bus controller (CBC), which performs block read/write operations on I/O devices over a fault-tolerant bus. To guarantee the *integrity* of bus operations, the clock and control signals are triply redundant, and are voted on by the I/O devices. The data and address buses are each augmented with a set of check bits, which are used for single bit error correction [6]. If one address bit is corrupted, the proper board and board channel will still be addressed. If a data bit is corrupted on the backplane during a write, it will automatically be corrected before it is applied to its output channel. To prevent bit corruptions in the first place, the signals are passed along a controlled impedance backplane, which reduces transmission line effects, and control / clock lines are interlaced with grounds to prevent signal crosstalk.

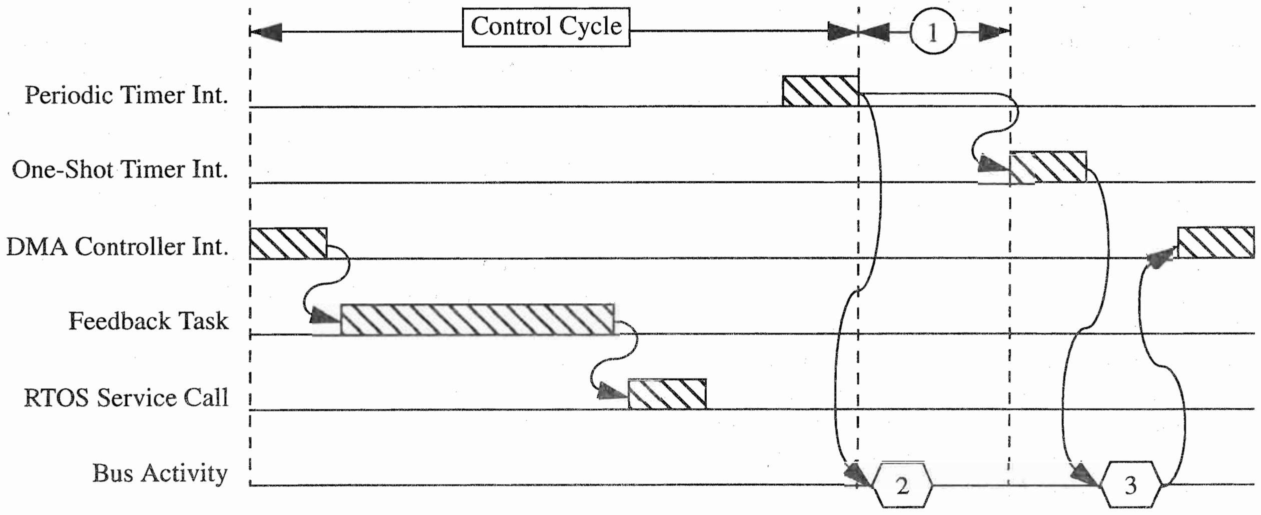
To maintain bus performance, decoding and encoding is performed by dedicated logic circuits on the bus controller and I/O devices. The check bit generation method for SEC is attractive because it lends itself to parallelization. When new data is presented, each check bit is generated concurrently by an independent combinatorial circuit. Read and write operations are pipelined such that the encode/decode operations affect the delay through the bus controller but not the bus throughput rate.

I/O Bus

The I/O bus, which operates at 4.0 MW/s and can address 512 I/O channels, has been designed with control applications in mind. For a regular bus address, the first six address bits specify the I/O board, and the last three specify the channel on the board. In addition, each board is assigned a unique "update" address. A write to the "update" address of an input board will cause all eight input channels to be sampled simultaneously, and a write to the update address of an output board will cause all eight output channels to be updated simultaneously. Besides the board-specific "update" address, there is a special address for sampling *all* input boards, one for updating *all* output boards, and one that samples *all* input boards *and* updates *all* output boards. The "update" addresses have been designed for multiple-input, multiple-output (MIMO) control applications, which may be sensitive to time skews in sampling inputs or updating outputs. The address which samples all inputs and updates all outputs can be used for establishing a precise single sample throughput delay, which simplifies modeling and analysis of the control system. With 8 channels assigned to each board and 10 slots available on the backplane, the controller can support up to 80 channels of I/O.

I/O Boards

The current system has four I/O devices: a D/A board, A/D board, a two-channel RS-232 link, and a JTAG interface. The 8 channel A/D board offers 14 bits of resolution and can support sampling rates in excess of 150 kHz. Differential inputs reject common mode noise and provide ground isolation between the controller and the sensors. Unlike many commercially available A/D boards which multiplex one converter across the board channels, this board has one sample-and-hold (S/H) device and one A/D converter for each channel. This permits all the channels to be sampled simultaneously and for all conversions to proceed in parallel. The 8 channel D/A board offers 14 bits of resolution, with differential outputs for common mode rejection and ground isolation. Temporary buffers hold new values as they are sent across the bus, and the new analog values appear on the outputs after a write to the board's update address. Low glitch converters help to reduce the transients which result from the conversion process. A programmable DUART on the mother board provides a two channel RS-232 link for host-to-target communications. One channel can be used for inspecting / modifying control variables and another for stream back data that may be captured by a host terminal. A JTAG interface (10.0 MHz serial link) on the motherboard can be used with a commercially available emulator for inspecting machine



(1) A/D Conversion Time, (2) Sample A/Ds & Transfer D/As, (3) Update D/As & Transfer A/Ds

FIGURE 3: Typical Control Cycle Timing Diagram

registers, single-stepping through code, and source-level debugging.

Real-Time Operating System

Support for the development of control algorithms on this hardware platform is provided by a real-time multi-tasking operating system. Tasks are assigned priorities, and the operating system always executes the highest priority runnable task. This ensures that time-sensitive tasks such as feedback controllers which are assigned high priorities meet their timing deadlines. Control schemes that have elements which operate on very different time scales can benefit from a multi-tasking implementation. This includes the adaptive unbalance response cancellation methods discussed in [8].

In the feedback control cycle illustrated in Figure 3, the operating system directs the sequencing of events and handles the low level hardware interactions. After new sensor data is ready for the feedback task, the task's context is restored, and it can begin immediately calculating the next set of actuator commands. These new output values are submitted to the operating system by a service call, along with a list of channels to update, and a time specifying when the update should occur. The service routine copies the values into a holding buffer, which contains a memory image of the next set of D/A values. The RTOS service call must be completed before the arrival of a periodic interrupt, which is generated by an on-chip event counter. This counter is clocked by an external 1.0 MHz oscillator that is common to all four processors. The periodic timer interrupt service routine (ISR) starts a one-shot internal timer, whose interval corresponds to the conversion time of the A/D converters. After that, it initiates a chained DMA transfer, by setting the start bit on the DMA global control register. This first transfer streams two block write bus operations out the appropriate communications port to the CBC. The first bus operation samples and begins conversion on all the A/D boards involved in a feedback control cycle, and the second transfers the contents of the holding buffer to the D/A boards. When the transfer out the communications port is completed, the DMA controller suspends itself, awaiting a signal from the RTOS. This happens when the one-shot timer fires, indicating the new A/D values are ready. The one-shot timer ISR executes and initiates the second and third DMA transfers. The second one updates all the D/A boards involved in feedback control, and when it is completed, automatically initializes itself (autoinitializes) for the third transfer, which retrieves new A/D values from the inbound communications port dedicated to the CBC, and places them in an RTOS holding buffer. The third transfer ends by firing a DMA interrupt, whose service routine copies the data from the holding buffer into a buffer specified by the feedback task, thus completing the control cycle.

This DMA intensive approach to servicing the I/O needs of the feedback task has a number of benefits. Foremost, the use of the DMA controller permits other CPU activities to proceed in parallel with I/O transfers. Keying the transfers off a common periodic interrupt keeps processors in lockstep by starting and stopping bus operations at the same time. Without a common clock, transfers would be keyed off of timers derived from local oscillators. Manufacturing tolerances in oscillator frequencies would cause program timing differences to accumulate, and exceed the 8 μS voter timeout window, ultimately causing a processor "failure". To further ensure that such an unnecessary re-configuration does not occur, this I/O transfer sequence occurs even if the feedback task does not request new sensor data. This periodic bus activity maintains synchronization between all four processors. This approach does require a certain amount of service overhead even when the I/O is not requested, but it is small because the chained-transfer is essentially a "touch-and-go" operation that demands little processor attention.

Another advantage of using DMA for transfers to (from) communications ports is that each word transfer can be synchronized with that port's output (input) ready line. This means that a new data item will be written to (read from) an output (input) port as soon as its ready line is asserted. This requires less time than methods which need more processor intervention. For example, the communications port could be configured to interrupt the processor whenever the output ready line is asserted. However, this causes a break in the 'C40's pipeline, and requires a context save/restore operation by the ISR. If the 'C40 is in a critical region, there will be latency between the time the interrupt is asserted and it is actually serviced.

Besides synchronizing the DMA controller and the respective communications ports, another way bus I/O transfers are made efficient is by requiring that the A/D and D/A boards involved in feedback control to form separate contiguous address spaces. In this way, only four bus operations are required in every control cycle because (1) sampling the sensor outputs requires only *one* block write (board update addresses are contiguous), (2) obtaining new sensor data is only *one* block read operation, (3) sending new actuator data to the D/A boards is only *one* block write operation, and (4) applying new analog values to the actuator inputs is only *one* block write operation (board update addresses are contiguous). By reducing the number of I/O bus operations every cycle, total setup time needed for block read and write operations is reduced. Finally, as many bus operations as possible are lumped in one DMA transfer to avoid the 3 μS needed for each autoinitialization.

To enjoy the benefits that DMA-based I/O transfers offer, a certain amount of care must be taken in the design of the RTOS and the application program. For example, the critical regions of the operating system must have execution times less than the 8 μS to avoid accidental timeouts caused by interrupt response latencies. To reduce the randomness of DMA transfer times, the DMA controller has been given priority over the CPU when the same resource is requested. In the application program, all of its variables should be initialized to avoid accidental data disagreements.

SUMMARY

Many magnetic bearing applications demand sophisticated digital signal processing technology and require higher reliability than that provided by a single processor system. The new digital controller at the Center for Magnetic Bearings strives to meet both of these needs in one powerful, flexible platform.

REFERENCES

[1] D'Addio, J.K., R.D. Williams, F.J. Keith, S.J. Fedigan. "Advanced Digital Controller Design for Magnetic Bearings." In *Proceedings of the Conference on Recent Advances in Active Control of Sound and Vibration*, p. 408-419, Blacksburg, Virginia, April, 1991.

[2] Keith, F.J., "Digital Control System Design for Active Magnetic Bearings," M.S. Thesis, University of Virginia, May, 1988.

[3] Yates, S.W. and R.D. Williams. "A Fault Tolerant Multi-Processor Controller for Magnetic Bearings." In *IEEE Micro*, p. 6-17, August, 1988.

[4] D'Addio, J.K. "An Integrated Magnetic Bearing Controller," M.S. Thesis, University of Virginia, May, 1992.

[5] Fedigan, S.J. and Williams, R.D. "An Operating System for a Magnetic Bearings Digital Controller." In *Proceedings of MAG '93*, p. 131-140, Alexandria, Virginia, July, 1993.

[6] Johnson, B.W. *Design and Analysis of Fault Tolerant Digital Systems.* Reading, Mass.: Addison-Wesley, 1989.

[7] Holloman,J.K. *Surface-Mount Technology for PC Board Design.* Indianapolis: Howard W. Sams & Company, 1989.

[8] Knospe, C.R., Fedigan, S.J., Hope, R.W. and Williams, R.D. "A Multi-Tasking DSP Implementation of Adaptive Magnetic Bearing Control." To appear in *IEEE Transactions on Control Systems Technology.*