SAFEbus™

Kenneth Hoyme and Kevin Driscoll
Honeywell Systems and Research Center
3660 Technology Drive
Minneapolis, Minnesota 55418

ABSTRACT

SAFEbus is slated to become the first standard backplane bus for commercial avionics. It has been accepted as the draft for ARINC Project Paper 659. This is a key component of the next generation of commercial avionics as defined in ARINC 651—the first generation to be highly integrated.

Honeywell is to supply an integrated avionics system—the Airplane Information Management System (AIMS)—for the Boeing 777, a new wide-body airplane. The 777, which is scheduled for initial delivery to United Airlines in May 1995, will be the first commercial transport to carry a highly integrated avionics system. Honeywell has designed an innovative backplane bus, called SAFEbus, to provide communications of all data among the Line Replaceable Modules (LRMs) in the Boeing 777 AIMS cabinets. The success of an integrated cabinet hinges on the backplane bus. It is the mechanism responsible for maintaining the space and time partitioning required to ensure that independent functions, which are sharing the cabinet resources, cannot adversely affect each other, even if the designs of one or more of the functions are faulty. A custom protocol is necessary since no existing standard bus was found to provide the fault tolerance or partitioning required for AIMS validation.

OVERVIEW

The SAFEbus interface logic consists of a Bus Interface Unit (BIU) ASIC, a Table Memory, an Intermodule Memory and Backplane Transceivers. This logic is paired to provide immediate fault detection and containment. The backplane bus lines are configured in a unique form of dual-dual redundancy that simultaneously provides high integrity and availability. (See Figure 1.)

The SAFEbus protocol is driven by sequences of commands stored in the Table Memories. The BIUs in every LRM on SAFEbus are synchronized to the same point in their respective tables and mechanisms are provided to attain synchronization

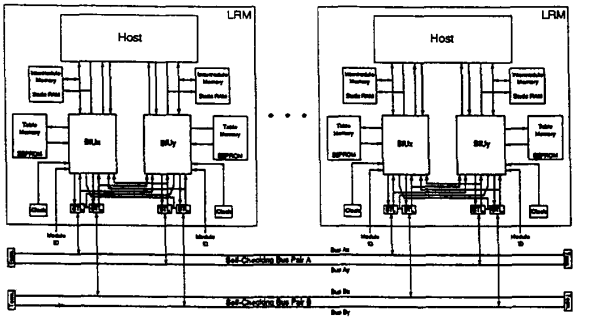

Fig. 1. SAFEbus Interface Logic

if it is ever lost. The bus time is divided into a set of "windows," each window containing a single message from 32 to 8,192 bits in length. The data is transmitted synchronously, two bits at a time, which is a drastic reduction in hardware over existing 16 bit, 32 bit or wider backplane busses.

Each command corresponds to a window and indicates whether its BIU should transmit or receive during the time assigned to that window. Each command also points to the Intermodule Memory location of the data to be either transmitted or received. The commands are organized into cyclic loops (frames) of constant length set by the sum of the individual window lengths. All the scheduling and addressing information is segregated into the Table Memory which is inaccessible to the cabinet's functions. This prevents a faulty function from affecting the timing or memory locations of other functions.

INTEGRATED MODULAR ARCHITECTURES

A revolution is occurring in the design of avionics systems for commercial air transports. The classic avionics system has a federated architecture. It consists of a set of functions, each of which is implemented in one or more Line Replaceable Units (LRUs). This federated architecture is being replaced by an integrated architecture that combines multiple functions into fewer LRUs. Integration offers many benefits, including: lower weight, lower power consumption, increased reliability, less frequent maintenance and greater flexibility. But precisely because functions share hardware resources, greater care must

SAFEbus is a trademark of Honeywell, Inc.

Based on a presentation at the 11th Digital Avionics Systems Conference, Oct. 5-8, 1992.
0885/8985/93 $3.00 © 1993 IEEE

be taken to ensure they will operate correctly, even if co-resident functions fail. This is the engineering challenge of integrated architectures.

An Integrated Modular Avionics (IMA) system differs from a federated system in that several functions are implemented in a single cabinet. Instead of a stand-alone box, the line-replaceable entity is an LRM, consisting of one or more circuit cards, that plugs into a cabinet. An integrated cabinet typically is larger than a federated LRU, but smaller than the sum of all LRUs that the cabinet replaces. At a minimum, the separate functions in a cabinet share a power supply and I/O units, and they may also share processing resources. To increase availability or integrity, functions can be replicated in multiple LRMs or in multiple cabinets. The attraction of the integrated architecture is the economies that can be achieved by sharing resources.

It is obvious that integrated systems need higher-performance microprocessors, denser memory technologies and higher-performance communications systems. But there is a subtler engineering challenge as well. Integration increases the risk that unwanted interactions among the functions residing on the shared hardware will lead to unforeseen failures. It is no longer sufficient to write an Interface Control Document (ICD) that defines explicit interactions occurring over separate communications paths. If an integrated architecture is to be successfully implemented, the complexity it presents to the certification process must be controlled.

The best approach to this problem is to attempt to make the execution environment of each function in the cabinet as much like the environment in the discrete LRU as possible. Essentially, all shared resources in the cabinet must be rigidly "partitioned" to ensure that one function cannot adversely affect another under any possible operating condition, including the occurrence of faults or design errors in the functions. Honeywell studies and work with the FAA have shown that strict deterministic control is the best way to ensure adequate partitioning.

Functions must be partitioned both in space and in time. Deterministic control over the partitioning of space means that it can be guaranteed that no function can prevent another from obtaining adequate memory space and that the memory space assigned to one function cannot be corrupted by the behavior of another function. Pre-allocated memory areas prevent contention for memory space. Hardware-based memory-protection mechanisms, such as processor memory-management units, are usually adequate to prevent corruption. Deterministic control over the partitioning of time means that it can be guaranteed that one function's variable demand for hardware resources will never prevent another function from obtaining a specified minimum level of service and, more importantly, that the timing of a function's access to these resources will not be affected by variable demand or by the failure of another function. If the system design does not build in time determinism, a function can be certified only after all possible combinations of events, including all possible combinations of failures of all functions, have been considered. Clearly this would drastically increase the cost of certification, as well as of software maintenance.

SAFEbus

The success of an integrated system hinges on the backplane bus. The backplane bus must be designed to support the dual requirements of space and time determinism. Our SAFEbus design also had to meet a number of additional requirements.

Boeing imposed an aggressive requirement for the number of days the 777 could be dispatched without maintenance following a failure. (The goal is to allow the plane to follow its normal schedule, which will eventually bring it to a maintenance base.) This requirement meant both that individual components of the AIMS system had to be reliable and that the system as a whole had to be fault tolerant. A second design requirement was that the backplane-bus interface not force complexity on the functions in an LRM. Some LRMs might be modern 32-bit processors, but others might be simple hardwired logic. A third requirement, implicit in the notion of an integrated cabinet, was that the design support a multi-processor architecture. In particular, the backplane had to provide adequate net throughput for the initial set of functions together with 50 percent extra capacity to allow for growth. Fourth, the integrity requirements for the avionics system as a whole meant that the backplane bus had to exhibit total fault containment. There had to be less than one chance in a billion per hour of operation that an error occurring within the backplane system would be passed undetected to application software. Finally, the design had to be one that would support the certification of the system and the re-certification of modified functions. In particular, the design could not be one that would force the re-certification of all functions when only one function was modified. Honeywell designed SAFEbus because no existing back plane bus met these requirements.

Protocol and Hardware

SAFEbus consists of two Self-Checking Buses (SCBs), A and B. Each SCB is itself composed of two buses, x and y. The interface logic, including the BIUs is also duplicated (see Figure 1). One of the BIUs transmits data on one of the busses in an SCB, and its partner transmits on the other bus. The data on any two busses which come from different BIUs are compared at the receiver. Only bit-for-bit identical data are written into the Intermodule memories. The receiving circuitry in the transmitting LRM also checks what is actually put on the bus for errors. Such self-checking ensures a babbling LRM will be detected and will remove itself from SAFEbus. This removal is enforced by having each BIU control the other BIU's drivers. If either BIU thinks it should not be transmitting, neither BIU can transmit. In general, the SCBs provide error-detection coverage that exceeds that provided by CRC codes, and they do so without consuming transmission time.

Using two SCBs provides immediate error correction for single-SCB transient errors. It also makes it more likely that the functions in the cabinet will remain available despite failures. SAFEbus is fail-operational/fail-passive: if one SCB fails, the cabinet remains in operation; if the second fails, the cabinet goes quiet.

All transmissions on SAFEbus are two-bit parallel. Each bus has two data lines and one clock line driven by the current transmitter. Such a narrow data path for transmission drastically reduces the total LRM pin count, increasing the system's inherent reliability. For the physical layer, SAFEbus uses backplane transceiver logic (BTL) that is an IEEE standard (IEEE 1194). The new logic has several advantages over the older TTL drivers, including lower driver capacitance and precision receiver thresholds. This allows incident wave switching for higher speed bus cycles. Other new backplane standards, including Futurebus+ and PI-bus, also specify it.

The bus time is divided into a set of "windows," each of which contains a message of from 32 to 8,192 bits (taking from 16 to 4,096 clock periods for transmission). The windows are separated by a small, fixed gap time, which is programmable to account for different LRM spacing and the total bus length. Two to four clock periods for the intermessage gap is typical. Messages that are to be transmitted or have been received over the backplane are placed in buffers in Intermodule memories. This organization permits a simple host interface, because the hosts view SAFEbus as a multi-port memory.

The SAFEbus protocol is driven by sequences of commands stored in the BIU's table memory. Each command corresponds to a a single message on the bus. The command indicates whether the BIU should transmit, receive or ignore the message. The BIUs are synchronized, so that at any given time all BIUs are at equivalent points in their respective tables. Mechanisms are provided to quickly attain synchronization should it ever be lost. The tables also contain the local address of the data to be transmitted or received. The commands in each BIU's table are organized into multiple frames. Each frame controls a repetitive sequence of windows which has a fixed total period. Under tightly controlled conditions, the bus may switch from using one frame to using another.

One of the benefits of the table-driven protocol is extremely high efficiency. Avionics applications typically generate short backplane messages, and most serial protocols perform poorly when messages are short. Efficiencies of between 10 percent and 30 percent are typical. The SAFEbus protocol, on the other hand, is over 89 percent efficient for a continuous stream of 32-bit messages. Because buffer addresses are kept in tables, they do not need to be transmitted on the bus. The use of transmit and receive commands in the individual tables eliminates the need to send source or destination LRM addresses. And because transmissions are scheduled, no transmission time is consumed arbitrating between contending BIUs. Except for the intermessage gap and the occasional synchronization message, all clock periods contain data. Thus a 60 megabit-per-second SAFEbus has a net throughput higher than 54 megabits per second. The backplane can be narrow rather than the wider parallel configuration of most backplane busses because the protocol is so efficient.

SAFEbus Determinism

The determinism of this design warrants more detailed examination, since no other backplane protocol provides it. Any protocol that includes a destination memory address in a message is a space-partitioning problem. It is extremely difficult to verify correct address usage in a partitioned multiprocessor. To ensure correct usage, the BIU would have to duplicate the processor's Memory-Management Unit (MMU) function. A difficult protocol would have to be implemented to ensure all BIUs used the same MMU information.

Any protocol that uses arbitration cannot be made time-deterministic. Arbitration is meant to ensure that when two LRMs contend for the bus, the one with the highest priority request is granted access. But minor jitter in the execution of functions can change which LRMs contend for the bus on any given bus cycle. As a result, the order in which the LRMs obtain access can vary from frame to frame.

SAFEbus achieves both time and space determinism by placing all message location and bus-timing information in the table memories. Fixed mapping of messages to unique locations in the Intermodule memory, which is protected by memory-mapping hardware in the host, guarantees space determinism. This table information is held in the BIU's table memory where it cannot be corrupted by any errant software or communications errors.

To make the system even more predictable, the execution of the software in the processing LRMs is synchronized with the execution of the commands in the bus table. Thus, the application software is at the same point during the same bus transmission window in every frame. One benefit is that message latencies are reduced; results can be scheduled to be transmitted just after they are generated, and data can be brought in from the I/O LRMs just before it is needed. A second benefit is that there is less latency jitter on cabinet outputs, which means that AIMS can be used in tighter control loops. A third benefit is that double buffering is rarely necessary because it is possible to schedule the transmission of a data block for a time when it is known the function software will not be accessing it or modifying it. The elimination of double buffers means the Intermodule memories can be smaller and memory access faster.

Synchronization of the bus schedule and the application-software's execution is guaranteed by embedding interrupt commands in the SAFEbus tables. On receiving an interrupt, the processor's operating system shifts to another application program. The interrupts take the place of the hardware timer other real-time executives employ.

The "arbitration" for SAFEbus occurs at design time, not run time. A software tool is being developed which parses a database of ICD information. This tool then generates the correct tables for each BIU on the bus. These tables are loaded into the BIU's Table Memories using an IEEE 1149.1 test bus.

SAFEbus Table Versioning

The SAFEbus protocol includes a mechanism to ensure that only LRMs with compatible tables synchronize with each other for normal operations. Two types of frames are supported. Unversioned frames allow LRMs of any SAFEbus version to communicate. Such frames always operate with the maximum programmable gap size between messages to allow LRMs from complete separate applications to communicate their version

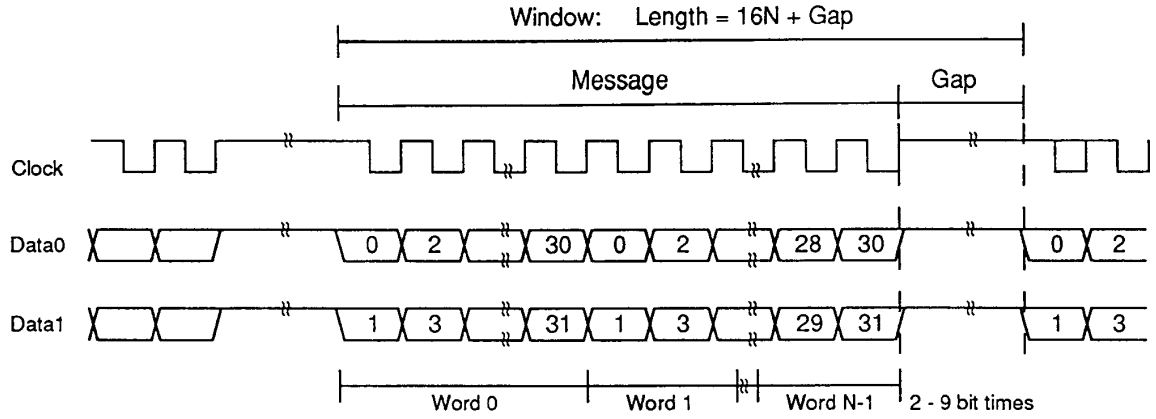

Fig. 2. Basic Message Structure

information with each other. After power on, all LRMs enter a standard unversioned initialization frame in which they transmit their versions so a cabinet software function can determine the correct configuration.

Versioned frames require all LRMs to be at the same version level. Frame Change Table Commands are provided to switch between frames. These commands inform all BIUs of the destination frame's version, or if that frame is unversioned. The mechanism which allows BIUs to regain synchronization also provides version information. Any BIU operating with a table of a different version drops off SAFEbus when a Frame Change or regain of synchronization message tells it that the bus is using a versioned frame different than its own. All normal flight operations are done in a versioned frame.

Slot location information is brought in to each BIU from the backplane connector to allow the BIU to verify that the table it is using is correct for the LRM's physical location. While each table contains a command for each window on SAFEbus, these commands are not bit-for-bit identical. It is essential that an LRM only operate with a table that is specific for the slot into which it is currently inserted.

SAFEbus Data-Message Structure

SAFEbus data messages have been designed to support the requirement of high efficiency. Because the protocol is table-driven, messages contain only data and do not include address and control information. There are two data-message types: basic and master/shadow. The basic message structure has been chosen to maximize the efficiency of data transmissions. The master/shadow structure supports data transfers by redundant or aperiodic functions.

Basic messages have a simple structure (see Figure 2). Each message consists of a string of one to 256 32-bit data words followed by a programmable intermessage gap (of two to nine bit times).

The master/shadow mechanism allows LRMs or applications to be reconfigured or spared without disturbing the traffic pattern on the bus. Master/shadow windows are identified by a field in the associated table command. As many as four transmitters can be assigned to one master/shadow window. Time-slot arbitration determines which of the transmitters actually gets control of the window. If the master is alive and has fresh data to send, it starts transmitting at the beginning of the window. The first shadow begins transmitting "delta" bit times into the window, but only if the master did not use its opportunity to transmit. The second shadow begins transmitting two delta bit times into the window, but only if the master and the first shadow did not use their opportunities to transmit. Finally, the third shadow begins transmitting three delta bit times into the window, but only if the none of the other candidate transmitters use their opportunities to transmit. Delta is a programmable value that is typically set at one bit time larger than the selected gap (values from three to ten bit times may be selected). The selected value depends on the propagation characteristics of the backplane. Unversioned frames use the maximum delta time of ten bit times. Examples of the transmission over SAFEbus when the master or third shadow transmits is shown in Figure 3.

Time-slot arbitration could re-introduce non-determinism, but strict measures have been taken to eliminate this danger. First, extra bit times in the window and a restriction on the size of the message guarantee that message transmission will be completed within the assigned time window, no matter what happens during arbitration. Thus, the time window remains the same size no matter which transmitter "wins" the arbitration. Second, recipients of a master/shadow message always place the data in the same memory location, no matter which transmitter wins the arbitration. Third, delta can be made large enough to guarantee that the candidate transmitters will never mistake a busy bus for an idle one and begin transmitting in error.

Synchronization Messages

The SAFEbus synchronization messages have been designed to support the requirements for integrity and time determinism. Cabinet synchronization is guaranteed in the face of any reasonable failure scenario, and synchronization does not require any centralized resource that could diminish the

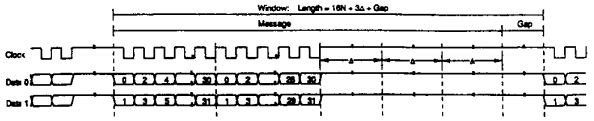

Fig. 3A. Master Transmits

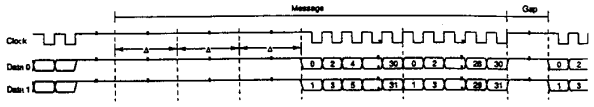

Fig. 3B. Shadow 3 Transmits
Fig. 3. Master/Shadow Message Structure

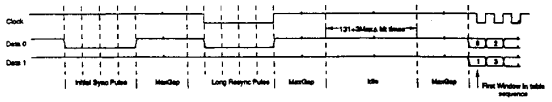

Fig. 4. Initial Sync Message

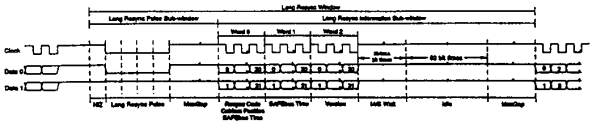

Fig. 5. Long Resync Message

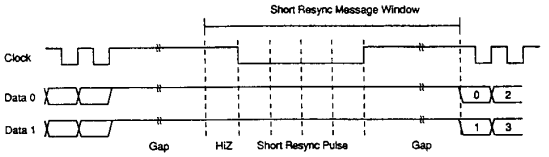

Fig. 6. Short Resync Message

system's integrity. Three types of synchronization messages are provided to bring the BIUs in step under three circumstances: at system reset; when an LRM is "lost;" and during normal operation.

The three types of synchronization messages are: the initial sync message (Figure 4), utilized at system reset; the long resync message (Figure 5) contains sufficient information to allow a "lost" LRM to regain synchronization with an active bus; and, the short resync message (Figure 6) used to correct for oscillator drift. Synchronization means that all BIUs are at equivalent points in their command tables. A resync code in the long resync message allows the BIU to determine which of 256 locations in the table memory it should jump to order to catch up with the other BIUs. Cabinet Position and Version information is used by a BIU to determine whether it is operating with a compatible table.

After any synchronization message is received, all BIUs are tightly synchronized (typically to within one bit time). Their oscillators would eventually drift, however, and thus close up the intermessage gap. Short resync messages are programmed into the command tables frequently enough to prevent gap closure.

To provide additional fault tolerance, all BIUs transmit the sync-pulse portion of every synchronization message. The multiple sync pulses are combined into a single pulse by the open-collector BTL drivers. Also, any LRM can originate an initial sync pulse. Because the synchronization mechanism is decentralized, no particular LRM must be operational to start up the backplane or to maintain synchronization.

Each BIU maintains a counter (called SAFEbus Time) driven by its synchronization-corrected oscillator. The synchronization mechanisms make the values in these counters the same in all BIUs. The time value may be used to time stamp data.

BUS Encoding

To improve error-detection coverage, data on the four SAFEbus serial lines are encoded in four different ways. Data on Bus Ax have normal polarity. Data on bus Bx are inverted. On bus Ay, every other bit is toggled, starting with the second bit. Bus By is the inverse of bus Ay. This is illustrated in Figure 7.

This encoding scheme allows bus shorts or transient upsets that affect several data lines simultaneously to be detected. It also allows quick detection of bus collisions caused by malfunctioning BIUs. Because bus lines are "wired OR," if a BIU-pair malfunctions and tries to transmit at the same time as another BIU-pair, illegal encodings appear as soon as the BIU-pairs transmit differing data. An additional virtue of this encoding scheme is that power consumption is independent of the data being transmitted. Two bus lines are always high and two are always low; when the data change, two of the buses change state and two do not. Because power consumption is constant, the power supply does not have to be designed for a worst case data pattern.

ARINC STANDARDIZATION

ARINC has recognized that the next-generation avionics systems will be integrated and has established a series of subcommittees to develop IMA standards for these systems. Subcommittees of particular interest include: the ARINC 651 integrated modular architecture subcommittee, which is establishing requirements and guidelines for cabinet-based integrated systems; the ARINC 650 packaging subcommittee, which is establishing standards for connectors and for LRM packaging; and the ARINC 659 backplane bus subcommittee.

The ARINC 659 subcommittee has been working for over three years toward the goal of defining a standard backplane

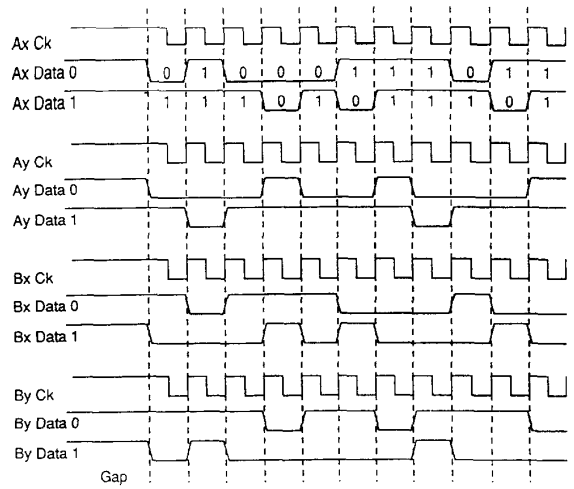

Fig. 7. Bus Encoding Example

bus for IMA cabinets. Although the requirements clearly pointed to a low pin count backplane, no existing bus had the required performance or integrity. After the award of the Boeing 777 AIMS contract, Honeywell submitted SAFEbus as a candidate standard, and it has now become the draft standard.

Although the requirements of AIMS can be satisfied by a clock speed of 30 megahertz, the SAFEbus BIU is being designed for higher speeds. Of course, the achievable speed depends on physical limits, such as the number of drops, their spacing and the total length of the backplane. Experiments with prototypes that include BTL parts show that backplanes as long as 42 inches containing 10 or 15 LRMs should be able to operate at speeds near 50 megahertz. However, current ASIC technology may limit the speed to the 30 to 40 megahertz range.

CONCLUSION

Data-bus standards are evolving in step with the avionics architectures they support. Integrated avionics architectures offer many benefits to airlines, significantly reducing their cost of ownership. But to be acceptable, the integrated systems must be as trustworthy as the federated systems they replace. In particular, there must be guarantees that the integrated functions cannot interact in unintended and undetected ways. The overriding goal in the design of SAFEbus has been to prevent such interactions and to control complexity so that airplane certification is straightforward.

The success of Honeywell's ATMs system on the Boeing 777 will undoubtedly lead to more highly integrated systems in future airplanes and thus to further evolution of the data-bus standards. But the goal of future efforts will be the same as today's: to improve system performance without compromising system integrity.

ACKNOWLEDGEMENT

The development of SAFEbus has been accomplished with the support of Honeywell's Air Transport Systems Division. The authors would like to express their appreciation to the many people at that division, and at Boeing Commercial Airplane Group, for their technical review and assistance. A particular note of thanks goes to the SAFEbus BIU design team for their diligent efforts and constant feedback resulting in many improvements to SAFEbus.

Kevin R. Driscoll received a BS in Computer Science from the University of Minnesota, Minneapolis, Minnesota, in 1986.

From 1971 to 1976, Mr. Driscoll was a cryptography specialist for the U.S. Army's Communication Command and Army Security Agency. He taught at the cryptography school in Fort Monmouth, New Jersey. In 1977, he joined Honeywell's Systems and Research Center where he currently is a Staff Scientist. He was a major designer of the PI-bus (the standard military backplane bus). He led the sub-committee which developed the requirements for the new military standard serial busses (HSDB). He led the effort to establish test, maintenance, and fault tolerance concepts for VHSIC and helped design the VHSIC TM bus which became IEEE 1149. He pioneered the use of self-checking pairs which is now a common fault tolerance technique, developed a fault tolerant fiber optic mesh communications systems, designed the first multi-tasking 1553 bus controller and designed the only ultra-reliable 1553. He has contributed to the electronics architecture design of: National Aerospace Plane (NASP), Space Defence Initiative (SDI), Light Helicopter Experimental (LHX), Boeing's 777 (AIMS and ADIRS), Advanced Launch System (ALS), Honeywell's vetronics program and Unmanned Underwater Vehicle (UUV). His current interests are real time and fault tolerant systems.

Kenneth P. Hoyme received the BS and MS degrees in electrical engineering from the University of Minnesota, Minneapolis, Minnesota, in 1979 and 1983 respectively.

He joined Honeywell's Systems and Research Center in 1983 where he is currently a Senior Principal Research Scientist. From 1983 to 1986 he was involved on various contracts related to the NASA Manned Space Station, inlcuding the development of advanced distributed control concepts for the life support systems. From 1986 to 1989 he was in charge of system architecture and software development for Honeywell's half-micron VHSIC signal processor. Since 1989 he has worked on development of architectures for integratred avionics systems, including the AIMS for the Boeing 777. His research interests include techniques for the development ofhighly-dependable systems for real time control.

Mr. Hoyme is a member of Eta Kappa Nu and Tau Beta Pi.