

A Firewire Based Data Acquisition System for Small Volume Positron Emission Tomographs

T.K. Lewellen, senior member IEEE, M. Janes, R.S. Miyaoka, member IEEE S.G. Gillispie,
B. Park, G. Herrmannsfeldt.
University of Washington Medical Center, Seattle, WA.

Abstract - Our laboratory has developed a data acquisition system (eMiCES) for small animal positron emission tomography scanners (PET) and other special purpose detector systems for radiotracer imaging applications. For our applications, the electronics had to be able to be mountable inside a rotating gantry. The resulting design consists of a series of modules that can be tailored for individual applications. The modules include a signal conditioning board (that can include analog signal summing), an analog processing board (digitization, integration, time stamping, and other pulse processing functions), and a communications board to the host computer via the 1394a bus (Firewire). The design takes advantage of the considerable commercial development of Firewire tools in the kernels of Microsoft and Apple computer operating systems. The various electronics modules also make extensive use of field programmable gate arrays (FPGA) for time stamping, pulse integration, and coincidence processing. Our current application of the system is for our small animal PET scanner (MiCES) with 72 detector modules organized as 36 Firewire nodes.

I. INTRODUCTION

The development of a small animal positron emission tomography (PET) scanner in our laboratory led to the definition of a series of requirements for a new data

acquisition system. The PET scanner (MiCES) consists of 72 detector modules. Each module contains a 6+6 cross anode position sensitive PMT (PSMPT) and an array of 484 0.8x0.8x10 mm MLS crystals. The detector modules are mounted on a gantry that rotates at a typical speed of 40° per second [1-6]. Thus, the data acquisition system had to deal with 864 analog signals from the PMTs and accommodate positional information from a rotating gantry. Further, the MiCES scanner is the first of what we expect to be a series of systems, and thus we decided to develop a data acquisition system that would easily adapt to future detector and gantry designs.

In order to reduce costs, we also wished to leverage existing technologies where feasible. We were unable to directly adapt existing commercial data acquisition systems to our needs, so we embarked on the design and construction of our own architecture – eMiCES. A major decision early in the development was to exploit the extensive support for high speed data transfer in standard desktop computers using the IEEE 1394 serial bus standard (Firewire). Table I lists some of the main characteristics of the 1394 standard. The 1394b standard was finalized after we had done our initial

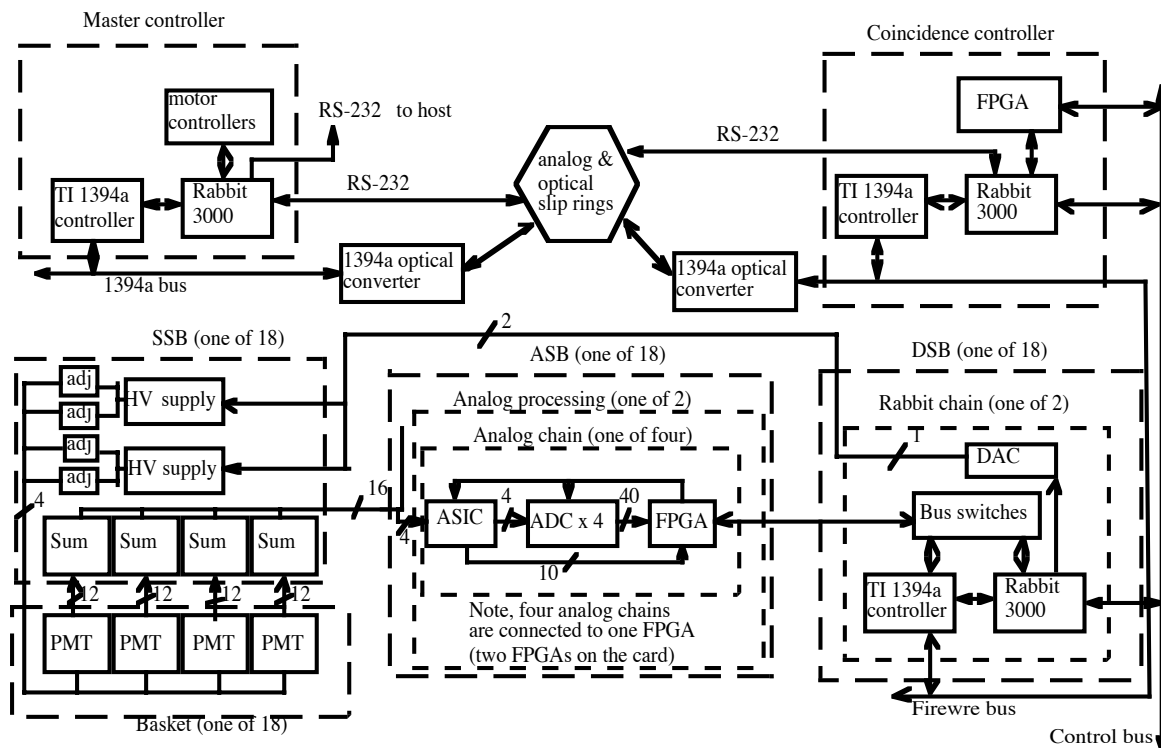


Figure 1: Block diagram of acquisition electronics system as applied to the MiCES scanner. The system can be adapted to a wide variety of detector modules and scanner geometries by changing the firmware in the FPGAs and local processor (Rabbit 3100 modules).

TABLE I
Maximum speed and packet size for 1394 (Firewire) protocols.

	Max speed	Max hop (STP cable)	Max hop (UTP-5 cable)	Max hop (fiber optic)	Max packet size
1394a	400 mbs	4.5 m	n.a.	n.a.	2048
1394b (copper)	800 mbs	4.5 m	100 m	100 m	4096
1394b (optic)	1600 mps (3200 proposed)	n.a.	100 m	100 m	8192

development work. Since the 1394a implementation is adequate for our MiCES scanner, we have not moved up to the 1394b option. As discussed below, there are other advantages to adopting Firewire including stability of the application programmers interface library routines (APIs), the ability to avoid writing kernel level drivers, and the variety of sophisticated integrated circuits to implement Firewire devices.

II. MATERIAL AND METHODS

A. System Overview

The current eMiCES system utilizes 3 major subsystems for each detector module: 1) a detector signal summing board (signal summing board - SSB); 2) a signal digitizing and data formatting board (analog system board - ASB); and 3) a digital control and communication board (digital system board - DSB). For a full PET scanner system, we add a coincidence controller board (CCB) and a master controller board (MCB).

The basic electronics structure for the MiCES scanners is illustrated in Figure 1 and illustrates how the eMiCES modules can be configured for a particular system. There are eighteen sets of signal summing, analog, and digital processing boards – one set for each of the 18 detector cassettes used in the MiCES scanner. Each cassette contains four MiCE2 detector modules which are powered by two EMCO C12N high voltage supplies – one for each pair of PMTs. In each of the board sets, the signals from the four 6+6 PSPMTs are routed to a signal summing board (SSB). The analog summing networks on the SSB convert the 12 PSPMT signals to 4 ‘position’ signals (X-, X+, Y-, Y+) using a

Table II
Data packet definitions for each detector event for the MiCES scanner

Parameter	Number of bytes	notes
Module ID	1	Bit 8 = 0, bit 7 = PMT_Id, bits 6-1 = module ID
Time stamp	4	62.5 mHz clock scalars
TAC data	1	8 most significant bits
PMT signals	8	12 bits per channel
Singles	2	8 bits per PMT, scaled by 256

The 16 bytes of the packet are stored in the 1394a controller FIFO when there is a valid coincidence event. 128 of these packets are packaged in each 2048 byte 1394a data packet when a detector node sends data to the host computer.

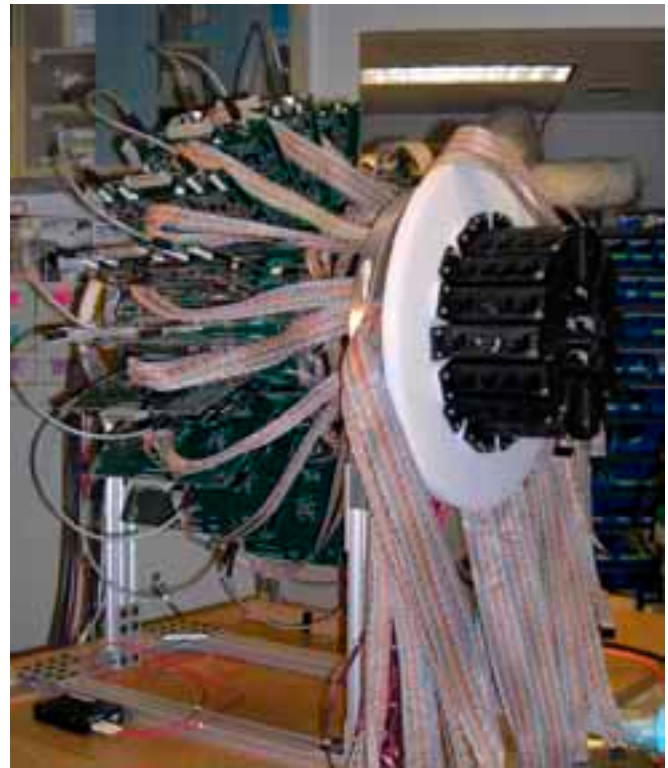


Figure 2: MiCES gantry (covers, SSBs and PMTs removed). The detector baskets and summing board mounting area are shown at the front, the analog and digital boards at the rear.

threshold-summing scheme implemented with high-speed operational amplifiers [6]. The board also includes two EMCO C12N high voltage power supplies – one for each pair of PSPMTs. The summing board approach for the MiCES scanner was adopted to reduce the number of electronic channels to be processed. The summed signals are then passed to the analog signal board (ASB).

The current ASB has three major components. The analog signals are processed by ASICs from Concorde Microsystems [Knoxville, Tennessee] (gain adjust and timing pickoff) and the analog data digitized by 62.5 MHz ADCs [3,7]. A pair of Altera [San Jose, California] FPGAs then processes the timing data and digitized analog signals. The FPGA code not only provides pulse integration and capture of the timing data, but also includes scalars to record the event time stamp and the trigger rate data. The FPGA includes first-in first-out (FIFO) buffers and the necessary logic to pass data onto the digital signal board (DSB). The DSB monitors a command bus from the coincidence controller card and commands sent via firewire from the host computer as well as packages the data into firewire packets for transmission to the host computer.

The coincidence card provides a coarse coincidence window for all events detected by the ASB cards in order to reduce the amount of data to be transmitted to the host computer. Currently, the rough coincidence window is set to 40 ns, but it is implemented in a FPGA and is easily changed. The coincidence board also provides the system clocks – a master clock (currently running at 62.5 mHz) and a sync clock (currently running at 1/8 the master clock rate). All of the

DSBs and the coincidence card use a Rabbit Semiconductor [Davis, California] 3100 module as the controlling processor. The MiCES scanner (with the detector modules and SSB boards removed) is shown in Figure 2.

The master controller, the only part of the acquisition electronics that is not mounted on the rotating part of the gantry for our current implementation of the MiCES scanner, communicates via a serial link to the host computer and provides master start/stop/reset commands to the other electronics subsystems. The master subsystem also controls the gantry and table motions. It utilizes a Rabbit 3100 as the controlling processor.

The host computer is currently a Macintosh Xserve but will be replaced by a G5 system in the near future. The software for the Rabbit 3100 processors is written in Rabbit Semiconductors Dynamic C and is identical for all 38 processors in the system. The code includes the ability to download into flash ROM new FPGA equations as well as provide local debugging services via a serial connection on each of the boards.

B. Control Structure

The basic control structure for the MiCES scanner implementation of eMiCES is shown in Figure 3. The host computer uses a simple serial port and the Firewire bus to communicate to the other components of the eMiCES system. The serial port is used for simple commands to the master controller such as position the table, set the acquisition time, start the scan, stop the scan, and reset the subsystems. In the MiCES scanner, the motion controller resides on the stationary portion of the gantry and communicates to the coincidence controller via a serial port. Table and gantry motion commands are handled directly by the master controller. Other commands dealing with the acquisition electronics are sent to the coincidence controller. For example, prior to starting the scan, the host instructs the master controller to advance the imaging table to the scan position, sets up the scan time, and starts the gantry rotation. When the start command is sent, the master controller passes it on to the coincidence controller. The coincidence controller asserts a start command on the command bus (broadcasts it to all of the DSBs). Each DSB resets their flags and sends a start command to each of the FPGAs in the eMiCES systems. The FPGAs wait for the next sync clock pulse to start processing data, assuring that all of the individual clocks used to produce the time stamps for each event start together. While the normally used set of serial commands is limited in scope, each Rabbit processor also will accept a series of “service” commands. In service mode, the Rabbit processor expects to be connected to a serial terminal and responds to a series of extended commands for exercising many aspects of the hardware being supported (e.g., downloading new code into the FPGAs, direct access to control and data registers in the FPGAs and the Texas Instruments Firewire chip sets). Thus, for both development and service tasks, each eMiCES board set can be operated independently via a direct serial connection, even when mounted in a full system such as the MiCES scanner. During normal acquisition each eMiCES node monitors the control bus and packages event data (Table II) that is sent to the host computer via Firewire.

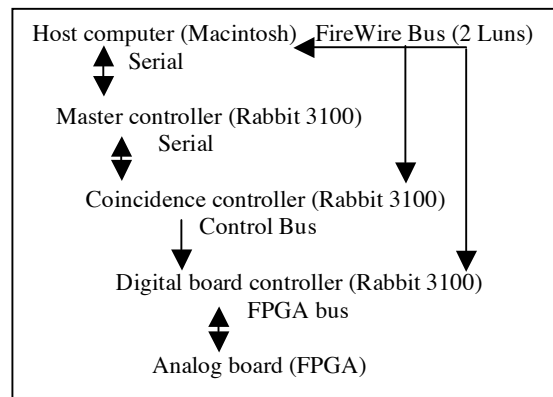


Figure 3: Topology of the control structure for the MiCES scanner. The serial bus is used to send simple commands to the master controller and coincidence controller. More complex commands are sent via the Firewire bus. The Firewire implementation provides two logical units (LUNs) in each device – one for data and one for control.

C. Firewire Implementation

Implementation of the eMiCES Firewire support involves two major development efforts: 1) a data acquisition software tool for the host computer; and 2) the hardware and firmware for the DSB board. The host computer software was developed using the Macintosh Xtools system with code written in Cocoa (a version of objective C) and C++. The code is organized as a series of library routines for all of the basic communication and data transfer protocols that are called by the main acquisition and service programs.

The host computer software takes advantage of several aspects of the Firewire standards. One of the major design elements was the avoidance of writing kernel level drivers and control routines. This was made possible by the way devices are discovered and assigned support by the operating system. At startup, the Rabbit processor loads configuration ROM data into the Texas Instruments Firewire controllers (TSB43AA82) used in eMiCES. That ROM data identifies each eMiCES node as being a SCSI device that uses the Serial Bus Protocol 2 (SBP2) standard with a non-standard instruction set. When a Firewire device is plugged in or powered up, the 1394a standard defines a signaling protocol that results in the host computer polling the devices on the bus and reading the configuration ROMs. In the Macintosh Os X operating system, our eMiCES devices are registered as independent nodes on the Firewire bus, but are listed as SBP2 devices (and thus able to respond to the SBP2 APIs). They are not assigned any of the standard system drivers (e.g., disk driver) since the nodes are configured as using a non-standard instruction set. This detail allows us to write all of our library routines in the user space – avoiding the necessity of writing low level drivers or worry about revisions of our routines for new releases of the operating system (the APIs definitions have been stable since the 10.1 version of the Os X operating system and we are currently running under version 10.4.1).

For general software development, the libraries we have developed have also proven to be very stable and it is a straightforward process to design new applications that utilize the eMiCES system. The libraries are self-configuring in

that they can support any number of eMiCES nodes. The initialization libraries provide lists of the available eMiCES nodes on the system and allow the application programmer to select any combination of the available nodes. During acquisition, a separate thread is spawned for each node, so that the system treats each node as an independent process with all processes reporting back to the main control routine. One concern about using a serial bus such as Firewire is the collision of packets (more than one node trying to use the bus at the same time). Fortunately, the SBP-2 protocol has built in logic to manage such problems including automatic retransmission and a queuing system between nodes to assure that one node cannot tie up the bus (giving equal access to all nodes). In our case, the Firewire hardware controller integrated circuits used in the eMiCES nodes contain the SBP-2 protocol logic on-chip. Thus, the software we generate does not have to take into account collision correction and many other aspects of the protocol.

Another aspect of the system design is extracting the maximum performance from a group of eMiCES nodes. Each data packet transmitted over Firewire incurs overhead in that such a transmission actually requires a series of packets to be exchanged between the host computer and the transmitting node. To reduce the percentage overhead, the maximum data packet size should be used. The maximum allowed packet size is a function of the Firewire speed (Table I). For the current eMiCES system, we are using the 1394a standard at 400 Mbs – resulting in a maximum packet size of 2048 bytes. That includes a basic header block which in turn reduces the actual “payload” size to 2016. Our data structure allows 128 events per data packet (Table II). Due to the packing of 128 events per node before transmission to the host, the time ordering of events is not preserved as they are stored on the host. We exploit that fact with our independent node task structure with each node process storing its data in a list mode file. After data acquisition is complete, the data from all nodes are re-sorted and the events put back in the proper time order using the time stamp recorded in the event packets (Table II).

To assure that the host computer can track any errors in the individual nodes, we programmed our eMiCES nodes to setup two logical units (LUN) for each node – a control LUN and a data LUN. In that way, the host process can request status information from a node and determine when it should issue a command to send data. A snapshot of a data transfer transaction is shown in Figure 4. While using a dual LUN architecture per node increases the bus overhead, it also increases the robustness of the system to detect and recover from errors. As shown in Figure 4, our implementation requires the host to query the control LUN to find out if data is ready before issuing a data read command to the data LUN. The result is a significant amount of time required to complete the sequence of events as the host computer processes the control LUN return data, sets up the data LUN read command and waits for the completion of the data transmission (typically ~ 200 to 400 μ sec with a 1 GHz host). However, since the bus is interleaving the data streams from all the nodes (37 in the MiCES scanner implementation), the overall data rate is still more than adequate for our requirements.

The eMiCES nodes utilize the Rabbit 3100 core module (Rabbit Semiconductor, San Jose, Ca.) as the controlling processor to load and control the FPGAs and the TI 1394a controllers. While the TI controller handles the major portion of the SBP-2 protocol automatically, there are several tasks that must be handled by the Rabbit. We currently use the Dynamic C compiler for the Rabbit for code development. Like the host code, the Rabbit utilizes multitasking with procedures watching the command bus, signaling the FPGA, and providing the needed support for the TI controller. Figure 5 depicts the major tasks that are done by the host, the Rabbit and the TI controller during a data read request.

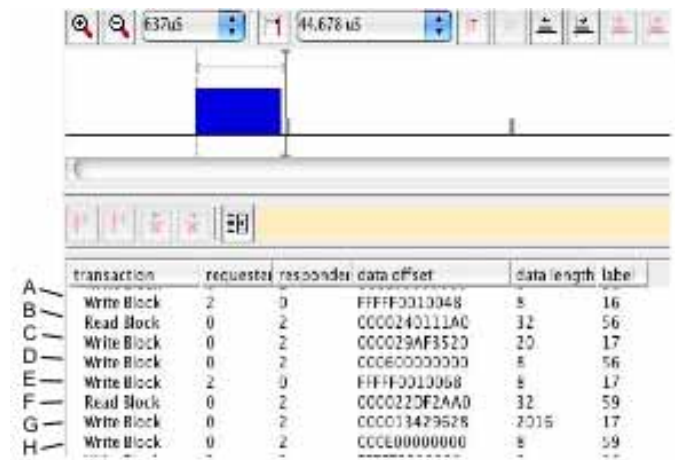


Figure 4: Image for a 1394a bus analyzer depicting a data transfer from a eMiCES node to a Macintosh 1 GHz PowerBook acting as the host computer. A: host to node control LUN request to send a command block. B: node read of command block data (is data ready?). C: return data to host as requested by command block (data ready). D: SBP-2 node to host status write (indicating transaction is done). E: host to node data LUN request to send a command block. F: read of command data (send data block). G: node to host transmission of detector data. H: SBP-2 node to host status block (transaction done). The top of the figure depicts the transmission time for the 2016 detector data packet (< 47 μ sec.).

Once an acquisition starts, the FPGAs begin acquiring data. When an event occurs on a DSB board, the FPGA integrates the pulses, loads the current run time clock into a register and adds to it the TDC data from the ASIC to form the time stamp for the event. While that is taking place, the DSB also asserts an event trigger on the event bus to the coincidence controller. If two DSBs generate event triggers within the coarse coincidence window and if the two DSBs are within the angular acceptance window defined in the coincidence board FPGA code, then the coincidence board generates an accept event pulse back to the DSBs. Otherwise the event data is cleared on the DSBs. Once an event accept pulse is received, the data is transferred to the TI controller built in FIFO. When 2012 bytes have been transferred, the DSB FPGA puts data into a FIFO defined within the FPGA and sets a flag for the Rabbit that the last 4 bytes of data are ready. This arrangement gives the Rabbit time to setup the needed return header information in the TI controller registers before the data block can be sent. When the TI registers are setup, the Rabbit signals the FPGA to send the last 4 bytes which triggers an automatic send sequence in the TI controller.

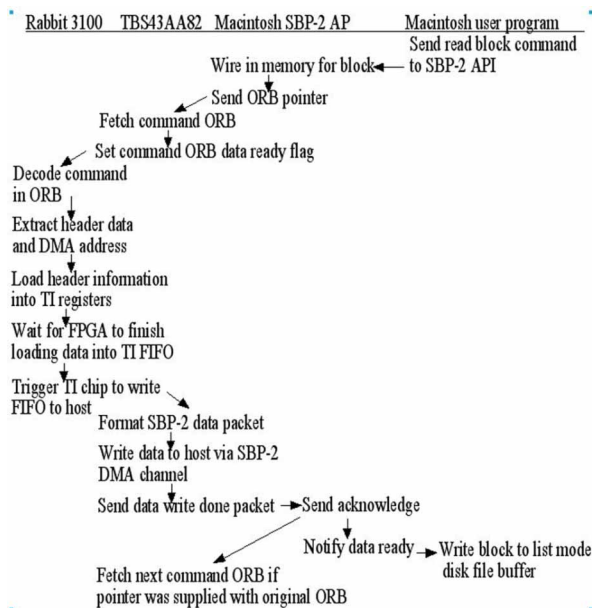


Figure 5: Flow diagram of the basic 1394a (Firewire) commands between the host computer and the data acquisition electronics for the MiCES scanner for sending a data block to the host computer.

The coincidence board also sends data to the host computer consisting of the current gantry position, any gating information (up to four channels) and the time stamp when the data was captured (typically set for once every 10 milliseconds). Of course, all of the parameters for both the ASB/DSB and coincidence board subsystems are changeable at run time via commands from the host to the master controller.

III. RESULTS

A full set of eMiCES modules (ASB, DSB, and coincidence board) have been assembled and extensively tested. A complete MiCES scanner is being assembled with 72 detector modules (18 sets of eMiCES ASB/BSD pairs plus the coincidence and master controller boards). Tests of the single set of modules indicate that confidence rates of up to 400 kcps will be supported for the MiCES scanner application. Full system tests and final performance measurements are being conducted in the summer of 2005.

IV. DISCUSSION

One critical part of the design is the time stamping of events by each of the ASB/DSB card combinations. Each FPGA maintains an event time scalar driven by the master clock. Care is taken to keep all of the clock cables from the clock fan out on the coincidence controller to each DSB the same length (minimizing clock skewing). The start sequence we have devised sends a start command over the command bus to all DSBs at the same time. However, the command bus is a daisy chain between DSBs and thus the command arrives at each DSB at a slightly different time. In addition, the Rabbit processors do not run off the master clock, and thus the various Rabbits will respond to the start command at slightly different times. The implementation of FPGA code that waits for the next sync clock pulse after receiving a start command

from its associated Rabbit processor solved the problem of getting all of the FPGAs to start their event scalars at the same time. The residual time skewing due to slight differences between ASICs, etc, is easily corrected with a timing calibration.

The current estimated maximum coincidence rate of ~ 400 kcps meets the requirements for the MiCES scanner implementation. Faster rates can be achieved by simply upgrading the DSB to the 1394b standard (a matter of replacing the main Firewire controller, the related clock crystal, and the Firewire connectors). Thus, we can double (via copper wire) or triple (with optical cable) the Firewire bus bandwidth with currently available controllers. Future projects also include a new SSB/ASB board combination that does not perform analog summing but rather provides simple gain adjustment and digitization of all signals and subsequent pulse processing via newer, high speed FPGAs. The DSBs include memory connected to the FPGAs so that look up tables for tasks such as crystal identification can be implemented. Once that is done, we can pack more events per data packet from the current 128 events (raw data) to 201 events (processed data – replacing the PMT data with the crystal ID and only passing data if the event is within the energy window). We also expect to implement more of the basic scanner commands via Firewire and eliminate the serial bus between the host computer and the eMiCES modules.

The eMiCES system is meeting all of our design goals and has considerable flexibility for our future needs. Designs for our next generation of SSB and ASB boards has begun. All of our designs (hardware and software) are being documented to allow others to adapt our designs to their needs.

REFERENCES

- [1] Lewellen, TK, M Janes, RS Miyaoka, SG Gillispie, B Park, KS Lee, and PE Kinahan. *System Integration of the Mices Small Animal Pet Scanner. Proceedings of the IEEE Nuclear Science Symposium and Medical Imaging Conference*, Rome, Italy, 2004.
- [2] Lewellen, TK, CM Laymon, RS Miyaoka, M Janes, B Park, K Lee, and PE Kinaha. *Development of a Data Acquisition System for the Mices Small Animal Pet Scanner. Proceedings of the 2002 IEEE Nuclear Science Symposium and Medical Imaging Conference*, Norfolk, 2002.
- [3] Lewellen, TK, RS Miyaoka, M Janes, B Park, and PE Kinahan. *System Electronics for the Mices Small Animal Pet Scanner. Proceedings of the IEEE Nuclear Science Symposium and Medical Imaging Conference*, Portland, Or, 2003.
- [4] Miyaoka, R. S., Janes, M., Lee, K. S., Park, B., Kinahan, P. E., and Lewellen, T. K. *Development of a Prototype Micro Crystal Element Scanner (Mices): Quickpet Ii. Molecular Imaging* (2004): In press.
- [5] Miyaoka, R. S., Kohlmyer, S. G., and Lewellen, T. K. *Performance Characteristics of Micro Crystal Element (Mice) Detectors. IEEE Trans Nucl Sci* 48, no. 4 (2001): 1403-07.
- [6] Park, B, TK Lewellen, RS Miyaoka, M Janes, and PE Kinahan. *Analog Processing Board for the Mices Small Animal Pet Scanner. Proceedings of the IEEE Nuclear Science Symposium and Medical Imaging Conference*, Rome, Italy, 2004.
- [7] Swann, B. K., Rochelle, J. A., Puckett, B. S., Bialock, B. J., Terry, S. C., Moyers, J. C., Young, J. W., Casey, M. E., Musrock, M. S., and Breeding, J. E. *A Coustom Mixed-Signal Cmos Intergrated Circuit for High Performance Pet Tomogrpah Front-End Applications. IEEE Trans., Nucl. Sci.* 50 (2003): 909-14.