A LOCAL AREA COMPUTER FOR DATA ACQUISITION AND CONTROL

Tomas Russ, Zdenek Radouch, Coles Sibley
MIT Bates Linear Accelerator Center
P.O. Box 846, Middleton, MA 01949

Abstract

A low cost control and data acquisition computer system was developed to be used as an element in distributed architecture control systems. It can be viewed as a low cost replacement for the CAMAC crate and its associated computer. This Local Area Computer (LAC) is based on the STD bus backplane, a low end industrial standard. Currently, it runs on a 8088 based processor card. Communication with other computers on the system is achieved through a specially designed SCSI-Ethernet controller, which can be programmed to allow sending of data only during a specified "time slot". This mechanism avoids delays on the network due to collisions, and enables the LAC to dump its entire "database" at regular intervals for retrieval by one or several hosts. Software -written in C and running under the VRTX real time executive- mimics the easy access to the hardware typical of CAMAC modules, allowing interfacing to control system software developed at other laboratories. Most of the LAC components are "off the shelf" items. However, some peripherals specific to accelerators, such as high stability voltage references and charge ADCs, were developed in house.

Introduction

Controls systems for accelerators have been following the general trend in the computer industry of distributing the processing and data acquisition functions. Given the sheer physical size of most accelerators, distributing the interface to the hardware so they can be in close proximity, can be justified often times just by the savings realized in cabling costs. In the past, such distribution was obtained by connecting several CAMAC crates to a controlling computer. Now a days, with the steep drop in the price of computer hardware, it is more economical to distribute not only the harware interface, but the entire computer system.

A typical control system architecture (Fig.1) can be thought of then as a series of data processing computers (workstations), data acquisition/control or front end computers — which interface with the hardware — and a data network that allows them to communicate.

Characteristics of Front End Computers

The front end computers are required to respond very fast to control requests from the data processing computers. They are also in charge of gathering all the accelerator related instrumentation data for use at the data processing end.

Most of the interface with the hardware in accelerator control systems can be implemented as simple reading or writing to I/O ports. Tasks such as setting a magnet or reading a BPM are normally reduced to writing a 16 bit number to a D/A converter, or reading the digital output of an ADC. In those cases, very little local processing power is needed. One can think of having the front end computer take care of some of the scaling or noise reduction tasks, but in general these jobs are better handled by workstations at the data processing end.

In other cases —perhaps 10% of the total hardware interface functions— it is clearly beneficial to add some on-board processing. Take as an example a system to measure beam profile by moving a wire accross the beam and reading the resultant secondary emission. This involves the coordination of moving the wire —perhaps with a stepper motor—, and simultaneously digitizing the secondary emission. A great number of points would have to be processed to obtain the desired result, which is normally a couple of numbers that described the best fit gaussian to the beam. Clearly, processing every point through the network makes little sense.

In yet other cases, perhaps intermediate in complexity between the simple I/O port access and the data reduction process of the beam profile monitor example, protocol conversions are needed to make the data accessed by the workstations meaningful. A good example would be the reading of GPIB devices through a front end computer. Typically, the user at the data processing end would like to receive the actual data read from the GPIB device. The ASCII sequence of control characters needed to produce this data should be hidden, handled by the front end computer.

The Local Area Computer

We have developed a front end computer that is designed to handle a subset of the total hardware interface in a typical accelerator control system. This subset can be a certain physical area (all the magnets, BPMs, current and vacuum monitors in a ring arc, for example) or a functional area (e.g., all BPMs).

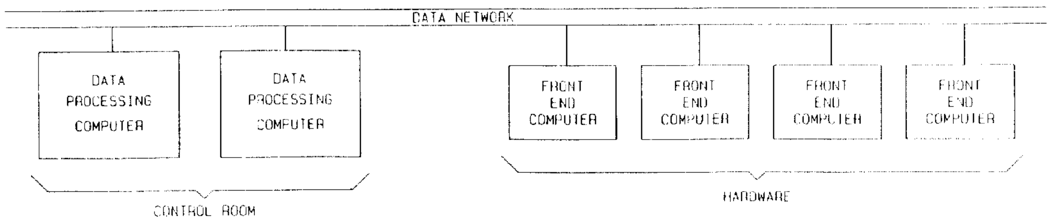

Fig. 1 Typical Distributed Architecture Control System.

These Local Area Computers (or LACs) can be accessed only through the data network (in our case, Ethernet). In their present version, they are meant to handle tasks that are not CPU intensive, such as direct I/O port access and protocol conversion. A consistent software interface has been designed to allow for different hardware configurations in the future.

The LAC architecture is based on the STD backplane, a low end, low cost, industrial computer bus [1]. As shown in Fig.2, the typical configuration consists of a CPU card (8088) with an incorporated serial port and interrupt controller, a memory card, and several peripheral cards to communicate with the accelerator hardware. Most of these cards are available from commercial suppliers. Some of them, specific to accelerator needs, were developed in-house[2,3].

The interface with the Ethernet network is done through an intermediate bus, the Small Computer Systems Interface (SCSI). This was done in order to allow different backplane buses to be used in the future without modifying the network connection. All known computer manufacturers support the SCSI bus (normally used for connection of mass storage devices).

LAC communication

As mentioned before, the two main tasks performed by these front end computers are to satisfy control requests from the data processing end, and to acquire and transfer instrumentation data. These two requirements have different urgencies associated with them. Control requests need to be satisfied as soon as possible (change the setting on a magnet, for example). General instrumentation data ought to be refreshed at an adequate rate, but does not carry as a whole that much sense of urgency. There is also a big difference in volume of data. Control requests normally involve very little data transfer, whereas the amount of instrumentation data can be quite big.

We have therefore established two different mechanisms to deal with these requirements. For control purposes, a request

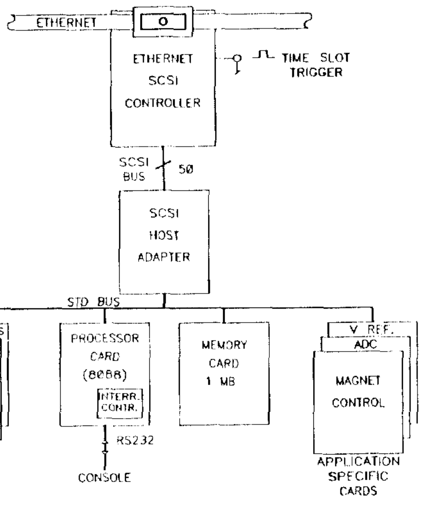

Fig. 2 LAC architecture.

packet is received though the network by the LAC. The LAC executes the desired action, and returns the packet to the originator with the reply status. This process takes typically less than 50 ms.

For instrumentation data transfer, an unsolicited transmission of all instrumentation data acquired by the LAC is periodically produced . This transmission is not meant for any destination in particular, but carries rather a multicast destination [4]. Any data processing computer interested in that particular piece of the instrumentation database can attach to that multicast address and receive the packet. The rate at which this transmission is produced is given by an external trigger connected to the Ethernet controller (Fig. 3). One typically sets it to 3 to 4 Hz.

This synchronic transmission of the instrumentation data accomplishes several purposes. On the one hand, the multicast addressing makes it possible to transmit only one copy of the instrumentation database, no matter how many data processing computers need it. On the other hand, the synchronism allows to set a different transmission time for each LAC on the network, therefore avoiding the possibility of collisions, which is the main source of real time response problems in Ethernet.

Synchronic transmission is not a normal feature of the Ethernet standard. In order to implement it, an in-house Ethernet controller-transceiver card was developed[5]. The trigger pulse, common to all LACs, is processed by an internal timer within the Ethernet card to insure that LAC transmissions are staggered. Notice that this feature in no way violates the normal Ethernet protocol.

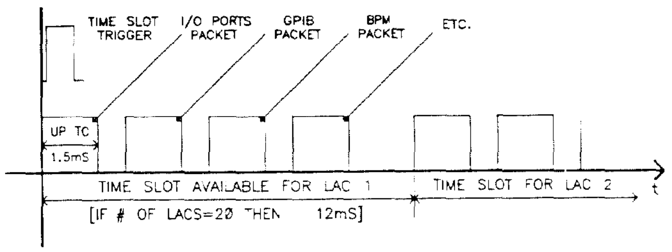

Fig. 3 Database Update. Every LAC multicasts all its instrumentation data (or the data that changed from last update) once every trigger pulse. A time slot is assigned to each LAC to avoid collisions on the Ethernet.

Network Access

All network accesses are produced at the data link level, i.e., all higher layers of the network protocols are bypassed. This significantly increases the speed at which transactions are made. One would think that the price is a loss of reliability in the network connection.

In a request/reply exchange, flow control (normally provided at the Transport Level) is not needed because of the straight acknowledgement of the request by the returned reply packet. Therefore a simple timeout at the originator end can be used to retry the request in case of failure. In a multicast transmission of instrumentation data, data may be lost. However, this isn't critical because of the periodic update of information. The addition of a sequence field in all multicast packets is used then to detect lost packets.

Software

Fig.4 shows the software diagram for a LAC. The SCSI driver and packet DEMUX are common to all LACs. Specific application drivers are constructed to interface with the hardware. Most LACs will have the two shown in the figure, IOP (standard I/O port driver) and GPIB.

In a typical request/reply sequence,a request packet is received by the SCSI driver, and analyzed by a port demultiplexer. The port number identifies which application driver this request is meant for. If that application exists, an empty buffer would be available in the corresponding queue, which is filled with the contents of the request and passed along to the application. The application (in this case IOP), executes the request and returns the packet to the originator through the SCSI driver.

In the case of a multicast transmission (Fig. 5), the SCSI driver receives an indication by the Ethernet controller that the last synchronic transmission has concluded (TRIGGER). All application drivers are then signalled. They, in turn, gather all the instrumentation data in a single packet and load it into the Ethernet controller for transmission when the next trigger pulse arrives. Each application transmits the packet with a different multicast address, allowing for the data processing computers to receive only the information they need.

All software for the LACs is written in C and runs under the VRTX real time executive[6]. Code is developed on an IBM personal computer and down loaded to the target for execution. Once stabilized, the intent is to burn it in ROM and have it run stand alone.

Measurements and Conclusions

Timing measurements taken on a LAC prototype are presented in Table 1. The data transfer speeds, however modest, are perfectly adequate for the needs of an accelerator control system.

LAC	
Request/reply	25 ms.
SCSI data transfer	>1 MByte/s.
STD data transfer	>150 KByte/s.
Jitter in time slot	<200 μs.

Table 1. Some Timing Measurements.

We described a low cost data acquisition/control computer that can perform most of the hardware interface tasks associated with an accelerator control system. Moreover, the use of standard data networks (Ethernet and SCSI), real time executive (VRTX) and high level language (C) open up the possibility of integrating other front end computers to the same network.

References

1. STD Manufacturers Group (STDMG), "STD Bus Specification and Practice", Doc.#10689E, Oct.1984.

2. A.Saab, "STD Compatible, 10 Bit, 8 Channel Fast ADC.", contributed paper T24 in this conference.

3. A.Saab, "STD Compatible, High Stability 16/18 Bit DC Reference.", contributed paper Y7 in this conference.

4. Digital Equipment Corp., Intel Corp., and Xerox Corp., "The ETHERNET, A Local Area Network, Data Link Layer and Physical Layer Specifications. V2.0", 1984.

5. T.Russ, "Ethernet Controller Adds Communications to SCSI Bus.", Electronic Design, **36**,20, Sept.8, 1988, 91–96.

6. Ready Systems, VRTX/86 User's Guide, Version 3, 1985.

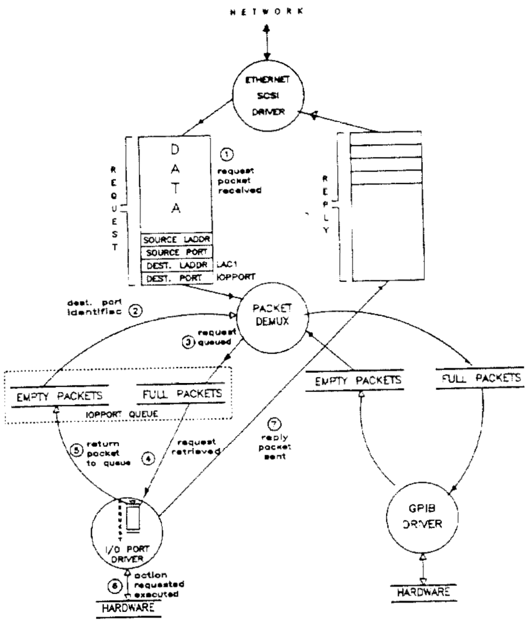

Fig. 4 Software Dataflow Diagram for Request/Reply.

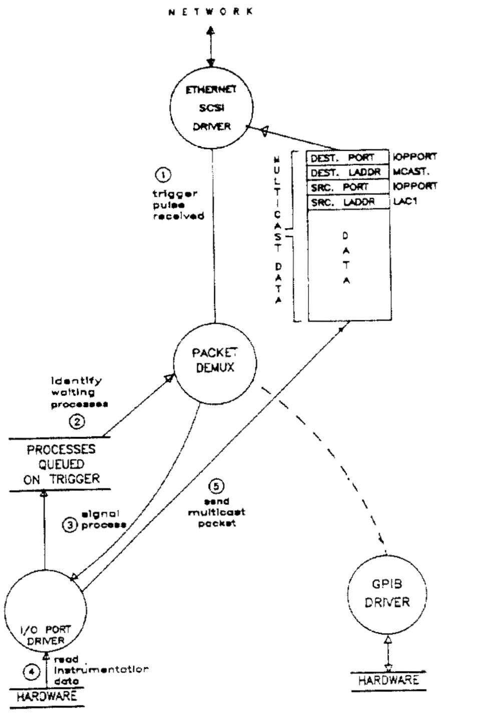

Fig. 5 Software Dataflow Diagram for Multicast Transmission.