Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

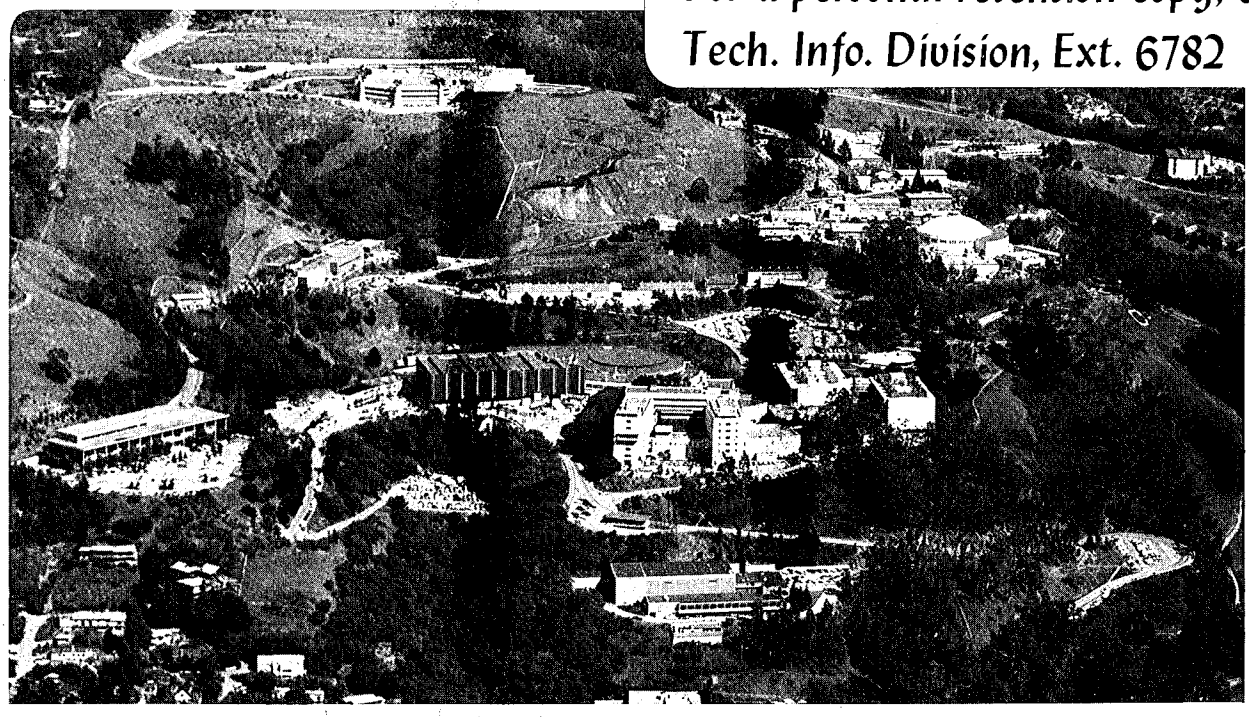
Engineering & Technical Services Division

To be presented at the IEEE Conference, Applications of Mini and Microcomputers, San Francisco, CA, November 9-12, 1981

USE OF EMBEDDED MICROCOMPUTERS IN SYSTEM DEBUGGING AND MAINTENANCE

John Meng and Dan Weaver

February 1981



Prepared for the U.S. Department of Energy under Contract W-7405-ENG-48

DISCLAIMER

LBL-12313

USE OF EMBEDDED MICROCOMPUTERS IN SYSTEM DEBUGGING AND MAINTENANCE

John Meng and Dan Weaver

Lawrence Berkeley Laboratory
University of California
Berkeley, California USA 94720

Abstract

Systems which use embedded microcomputers to perform control functions can often double as flexible debugging and maintenance devices by switching in high level language ROMS. This paper describes systems in which such benefits are successfully exploited.

Introduction

For several years we have been developing small systems using microcomputers as the primary system "intelligence". These systems are all similar in that they are dedicated to a simple combination of mutually exclusive functions in a slow or non-critical time sequence. After some early experimentation with tailor-made languages, we finally settled on using BASIC to implement the programs used in these systems. In general, they all have mechanical devices to drive (stepping motors or DC motors) and perform sundry other tasks such as reading pushbuttons, turning lamps on or off, and communicating with an operator.

Early in the development of these systems, the value of having a conversational interactive device through which we can probe and manipulate system elements became apparent. Critically valuable test loops can be constructed and executed in minutes, often saving hours of tedious work with an oscilloscope and "lashed up" test circuits.

We have since moved on to much larger systems. The latest is MIDAS (Modular Interactive Data Analysis System). The MIDAS prototype, consisting of a subset of the final system, is currently undergoing an evaluative testing program. Microcomputers pervade MIDAS as well as its many appendages, and these embedded micros serve as examples of the value inherent in the use of a high-level conversational language during debugging and maintenance phases.

MIDAS is a pyramid of processors (Fig. 1). At the peak is a single elaborate processor used to converse interactively with an array of users. In the midsection of the pyramid are arrays of mini-computer central processors--"super-micros"--operating in parallel and used to sort through a massive data base existing in an array of magnetic tapes, multimegabyte memories and disc packs. Microcomputers riddle the structure, serving as low speed switches and simple-minded decision makers in everything from the handling of mechanical sequencing in the disc and tape cpntrollers through handshaking with users off the peak of the pyramid.

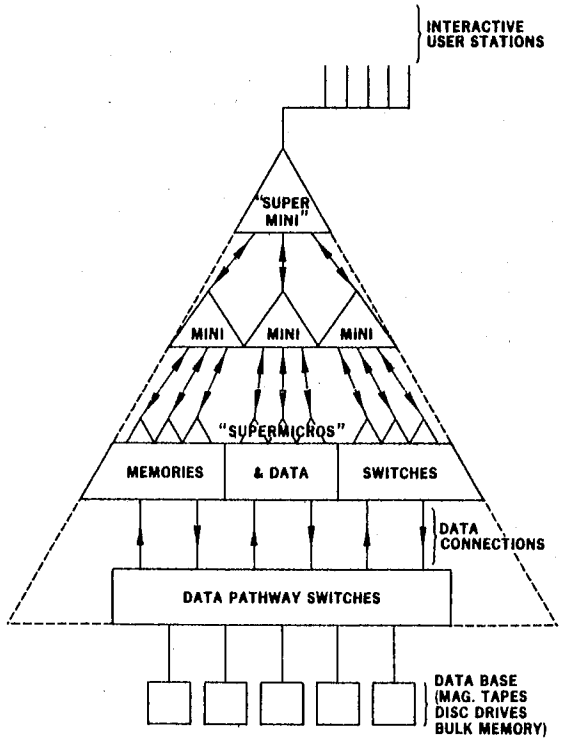


Figure 1. MIDAS

Embedded micros access the system at its most intimate level; for example, at the mechanical functioning of disc and magnetic tape drives. They also have privileged access to most data storage and data passageways. For example,

micros control settings of high-speed data pathway switches and physically carry out commanded switching of memory subsystems from one "supermicro" to another. Most places where the system is capable of "bending;" the joints where flexibility is effected, are under the direct supervision and control of microcomputers. In addressing both initial debugging and subsequent maintenance in the system, we are using the embedded microcomputers as frontline interfaces between engineering and/or maintenance personnel and many critical states and flows within the structure. To enhance this communication we have turned away from the computer-on-a-chip micro in favor of a slightly more elaborate configuration capable of executing short routines written in a high level language (BASIC).

Economics of the Expanded Micro

Microcomputers are used freely throughout MIDAS. Effectively, the microcomputers have become just another component. They are plugged into the system for reasons not philosophically dissimilar to those used to select other logic components, such as integrated circuits. Since we desire to incorporate these devices into our system based on their efficient usefulness rather than based on their cost, it is inherently vital that their individual and aggregate cost not make the system prohibitively expensive. Some additional expense is justifiable on the basis of savings in initially making the system operational (debugging) and on the basis of savings in reduced down-time and in reduced numbers of maintenance hours required to repair the unit following component failures (maintenance).

Specifically, traditional component-level maintenance involves the temporary electrical connection of a monitoring device and/or a tester into suspect regions of the system (Fig. 2). The failure mode is then hopefully reproduced by some operational acts and/or tester functions, often necessarily involving the entire pyramid, and the results are displayed in such a manner that the flawed logic becomes visible, allowing defective components to be replaced. Under ideal conditions, these procedures produce results quickly and efficiently. The assumption implicit in the "ideal conditions" are:

(1) The failure or flaw is explicit enough to be localized to the correct module and to the correct part of the module by its symptoms.

(2) The module under suspicion is physically easily accessible to the probes of test and monitoring equipment.

(3) The failure is "hard." That is, it is reproducible by a simple duplication of some part of whatever procedure brought it to attention initially.

As any experienced maintenance person can verify, the fulfillment of all these assumptions for any particular problem is rarely realized. Memorable problems requiring the uncanny deductive powers of a modern-day Sherlock Holmes seem remarkably common. This becomes more the case as the system becomes more complex.

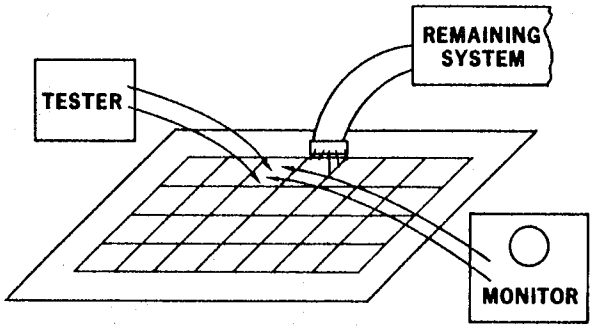


Figure 2. Traditional component-level maintenance

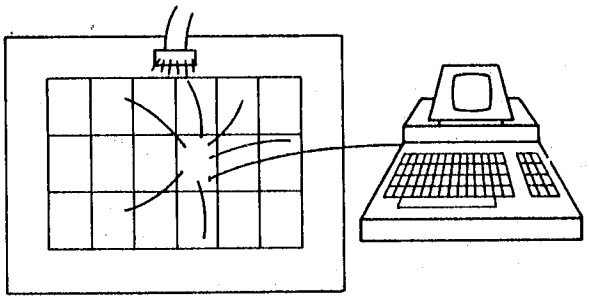


Figure 3. The embedded micro is connected to a terminal to allow maintenance personnel access to selected system components.

Micros embedded in the system can help in the realization of these ideal assumptions, significantly reducing the costs associated with debugging and maintenance. Specifically, each micro is local to a part of the system. When loaded with a BASIC interpreter, it becomes a subsystem independent of the remaining system. The disc controller, for example, contains micros to handle error correction. The disc controller contains three data ports. Each port contains an error-correction micro. Loaded with BASIC, any one of the three micros can exercise and diagnose failures in its own port while the remaining two ports are operating normally. Exercising subsystems independently adds a weapon to the diagnostic arsenal of the maintenance or debugging person. The ability to diagnose and localize failures is greatly enhanced by being able to exercise specific parts of the system independent of adjacent parts.

Embedded micros have "wired-in" access to intimate connections within the system. By connecting a single cable from an input/output port

to a terminal, the person doing the debugging gains conversational access to the system's intimate innards. Local loops can be composed and run on the spot to handle many types of testing.

Finally, the embedded micro can often help in finding "soft" failures. Local loops can be set up and left running in attempts to pinpoint such intermittent malfunctions, while the remaining system is left to run normally. Embedded micros capable of running in conversational modes with maintenance personnel provide convenient, rapid access to many internal parts of the system. They also often allow the major parts of the system to function while a local subsystem is being repaired. Savings both in the time required to repair subsystems and in the time during which the entire system would normally have to be dedicated to maintenance justify the expense of expanding the capabilities of embedded microcomputers.

Embedded Micros as Maintenance Devices

Our embedded micros are dedicated to system operations under normal circumstances. They incorporate memory-mapped connections and standard input-output connections to various gates and control signals (Fig. 3). When problems appear, the states of various control signals become important to the maintenance person. It is also often necessary to allow him to arbitrarily control the various gates and control lines to which the micro has access. One approach to accomplishing this is to assembly-language program the micro with dedicated maintenance aids. By invoking such aids either with a terminal or with a switch-box/ light-array combination, access is gained to prespecified system values and functions. A problem with this approach is the inherent lack of flexibility resulting from being required to "pre-guess" everything required for maintenance communications and to "pre-program" these requirements into the micro when the system is first assembled.

Most micro systems have debugging software available which allows maintenance personnel to load and run short routines from a terminal. Maintenance personnel familiar with microcoding techniques may find this valuable. Our approach, however, has been to remove the system's execution ROM's, the program used during normal operations, and to replace them with ROM's containing a BASIC interpreter. Maintenance routines are then loaded as BASIC source programs either from a terminal's keyboard or from a library on cassette tape. The programs are, for the most part, universally readable and can be readily modified to help entrap perverse failures thought up by Mother Nature's gremlins.

The embedded micro has not replaced the oscilloscope or logic analyzer as a maintenance tool. However, it can significantly enhance the use of these valuable tools by printing logic states at various crucial points without requiring direct physical access. It can also be programmed to repetitively exercise states over which it has control, providing signals to be viewed with the oscilloscope or logic analyzer.

Figure 4 shows the micros in the MIDAS disc controller. The main control micro has access to head-select and drive-select lines and to head-stepping signals. This micro also monitors both the sector number being counted from pulses from the drive and the sector number read from the data track. It is the device which begins a read or write operation when the correct position is reached on the disc. Malfunctions in the drive itself or in its self-contained electronics are easily probed by swapping the BASIC EPROM set into the main control micro. Positioning and read or write commands can be issued in loops to provide signals for viewing on oscilloscope or logic analyzer. Simple programs can be entered and run for checking the electromechanical functions of the mechanism.

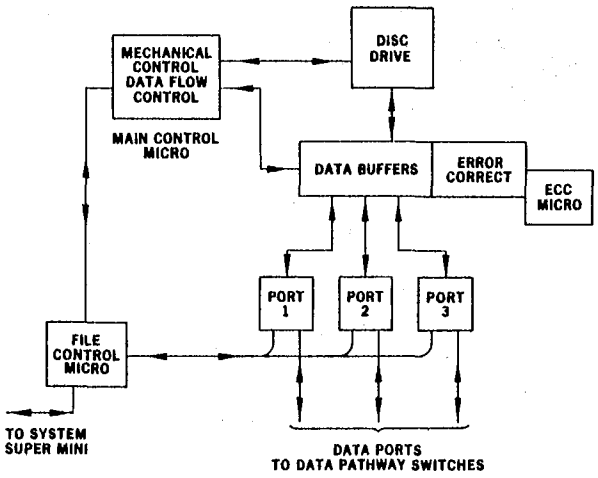


Figure 4. MIDAS disc controller

The ECC micro normally is invoked only when correctable data errors are detected. It has access to the data in data buffers as well as to ECC data. If problems arise in this area, BASIC programs can be loaded to run diagnostics on all the data storage in the controller. Loops can be run to provide test signals for viewing on oscilloscope or logic analyzer. The file control micro controls the disc connection, via ports I, II and III, to the remainder of MIDAS. It also issues global commands to the main control micro. Problems in these areas can be diagnosed and solved using techniques analogous to those associated with the other two micros.

Thus we see that each micro has its discrete region of responsibility and control. Using each as a debugging tool allows us to treat each discrete region independently. We do not have to involve the entire pyramid in a debugging operation on one of its internal components. We could

digress into a description of each of MIDAS's many embedded micros. However, the debugging and/or maintenance value of most are similar. Each allows us to divide, communicate with, and eventually to conquer problems in an efficient manner.

Another example, one with unique features, is the "conductor:" the subsystem which switches memories from "super-micro" to "super-micro." The controlling micro has no mechanical device to control. (Figure 5) This subsystem is labeled the "conductor" because it is also responsible for directing pre-orchestrated flows of data through the system processors. The conductor's players are FIFO's (First In, First Out, stacks) loaded with numbers (control bytes) representing memories. Each byte turns on gates connecting a processor to a memory. A pair of FIFO's is dedicated to each processor. One holds bytes corresponding to memories containing data to be processed. The other holds memory numbers corresponding to data already processed. The control micro passes numbers from "processed data" FIFO's into "ready to be processed data" FIFO's. Lists of numbers are initially passed to the control micro from a processor at the next higher level in the pyramid (the "Secondary CPU").

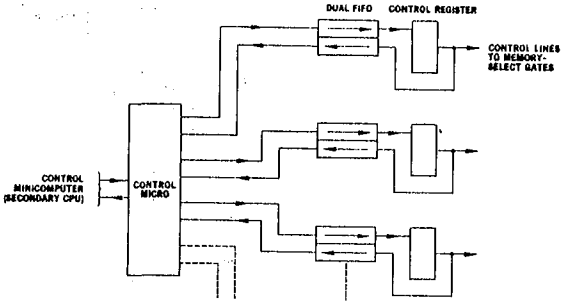


Figure 5. The MIDAS conductor directs data flows through the system processors.

The control micro, in the event of memory or processor failure, can be loaded with a BASIC interpreter, giving the service person access to FIFO controls. The maintenance person can then connect any memory to any processor for checkout and debugging. In this case, the embedded micro is not doing the actual checkout. It is simply giving the service person a convenient handle for manipulating parts of the system to expedite maintenance and checkout, but not requiring him to use remote elements of the system for his manipulations.

In summary, the precise use to which the embedded micro can best be put to expedite maintenance dependends on the position of the micro and the structure and nature of the subsystem into which it is embedded. However, in general it allows the exclusion of much uninteresting, functioning hardware from consideration during a maintenance pass. It also expedites communications between the hardware of interest and the person working on the hardware.

Developing a Cost Effective Micro

As was discussed earlier, we felt it desirable to enhance our embedded micros to where they would be able to run BASIC when necessitated by maintenance requirements necessitated this. On the other hand, over-enhancement could make the cost, both in money and physical space, a prohibitive factor in many systems. The unit we have built and are using requires less than 2-1/2" x 5" of wire-wrap board space and costs about $100 to build. (See Fig. 6). It is a reduced version of a commercially available 16-bit microprocessor system.

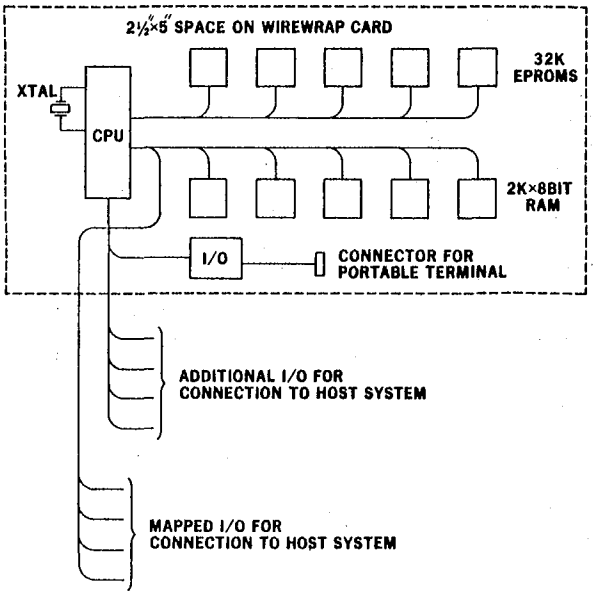


Figure 6. An embedded microcomputer module for use in MIDAS. BASIC is run in the micro by exchanging EPROMS.

Two versions of the microprocessor are produced by the manufacturer. One is a 64-pin device. The other is a 40-pin device. The major difference between the two is sixteen pins dedicated to the data bus on the 64-pin device versus 8 on the 40-pin chip. In order to make the module as small as possible physically, we chose the 40-pin device. The various problems we encountered as a result of this choice are discussed at length in Reference 6.

When running BASIC, the micro has about 1000 bytes of RAM available for source statements and variables. Figure 7 is a BASIC program (a memory diagnostic) which uses most of the available RAM.

It is included to give the reader a feel for the size of the program which can be run.

```
 10     GOSUB 1000
 20     BASE 0120H:: CRF[0]=05FFH:: BASE 02A0H:: CRF[6]=03EH:: CRB[6]=1:: CRB[7]=1
 30     GOSUB 1000:: BASE 0120H:: CRF[0]=01FFH:: GOSUB 1000
 33     BASE 0120H:: CRF[0]=05FFH:: GOSUB 1000
 40     DAT=0180H:: GOSUB 1080
 50     INPUT "ADDRW ";DAT:: IF DAT>999999 THEN GOTO 200
 55     IF DAT>99999 THEN GOTO 220
 56     IF DAT>65535 THEN GOTO 110
 60     CMD=0578H:: DAT= LNOT (DAT):: GOSUB 1010
 70     INPUT "DATA ";DAT:: IF DAT>65535 THEN GOTO 110
 80     DAT= LNOT (DAT):: CMD=057DH:: GOSUB 1010
 90     INPUT DAT:: IF DAT>65535 THEN GOTO 110
100     DAT= LNOT (DAT):: CMD=057EH:: GOSUB 1010:: GOTO 90
110     INPUT "ADDRR ";DAT:: IF DAT>65535 THEN GOTO 50
120     DAT= LNOT (DAT):: CMD=057BH:: GOSUB 1010
130     GOSUB 1040:: DAT= LNOT (DAT):: PRINT #DAT;DAT
140     INPUT DAT:: IF DAT>65535 THEN GOTO 50
150     CMD=0577H:: DAT=-1:: GOSUB 1010:: GOTO 130
200     INPUT "STRT ADDR ";DAT
210     CMD=057BH:: DAT= LNOT (DAT):: GOSUB 1010:: CMD=077FH:: GOSUB 1010:: GOTO 220
215     CMD=057FH:: GOSUB 1010
220     INPUT "REG ";DAT:: IF DAT>4095 THEN GOTO 50
225     GOSUB 1080:: GOSUB 1040
230     PRINT # LNOT (DAT); LNOT DAT
250     GOTO 220
1000     BASE 02A0H:: CRB[8]=0:: CRB[8]=1:: RETURN
1010     BASE 02A0H:: CRB[6]=1:: CRB[7]=1:: CRF[6]=03EH:: BASE 0120H
1015     CRF[0]=057FH:: GOSUB 1000
1020     BASE 0120H:: CRF[0]=DAT:: BASE 02A0H:: CRB[7]=0:: GOSUB 1000
1030     CRB[7[=1:: BASE 0120H:: CRF[0]=CMD:: GOSUB 1000:: RETURN
1040     BASE 0120H:: CRB[-16]=1:: CRB[-1]=0
1050     BASE 02A0H:: CRF[6]=03EH:: CRB[6]=0:: CRB[7]=0:: CRB[8]=0
1060     BASE 0120H:: DAT=CRF[0]:: BASE 02A0H:: CRB[8]=1:: CRB[7]=1:: CRB[6]=1
1070     RETURN
1080     CRF[6]=03DH:: BASE 0120H:: CRF[0]= LNOT DAT:: GOSUB 1000:: RETURN
2000     INPUT DAT:: CMD=056FH:: GOSUB 1010:: GOTO 2000

SIZ
PRGM:0390H BYTES
VARS:04H BYTES
FREE:072H BYTES
```

Figure 7. Memory diagnostic program showing approximate size of program which can be loaded and run in the embedded micro.

Conclusion

Embedded microcomputers can be enhanced sufficiently to be able to run BASIC during maintenance and debugging operations. The enhancement can be minimal enough to keep the cost in both money and physical space allocation within acceptable bounds.

The value of being able to execute BASIC in embedded microcomputers during maintenance and debugging operations is the ease with which the maintenance person can gain access to and communicate with vital signals in the system. The embedded micro allows system maintenance to be carried on within discrete subsystems without the necessity of involving other major parts of the system.

Acknowledgment

This work was supported by the Director's Office of Energy Research, Office of High Energy and Nuclear Physics, Division of Nuclear Physics and by Nuclear Sciences of Basic Energy Sciences Program of the U.S. Department of Energy under Contract W-7405-ENG-48.

References

On MIDAS:

1. Maples, Creve C., "A Specialized, Multi-User Computer Facility for the High-Speed, Interactive Processing of Experimental Data." Proceedings of Computerized Data Acquisition Systems in Particle and Nuclear Physics Conference, Santa Fe, NM, May 14-17, 1979.

2. Maples, Creve C., Proposal for a High Speed, Interactive Facility for the Reduction and Analysis of Scientific Data. Presented at Asilomar, CA, meeting of the American Physical Society, Nov. 1-3, 1978. LBL-7196 (1978), Lawrence Berkeley Laboratory, Berkeley, CA 94720.

References 3-5 presented at Topical Conference on Computerized Data Acquisition in Particle and Nuclear Physics, Oak Ridge, TN, May 28-30, 1981. All appear in Proceedings of the conference.

3. Maples, C., Rathbun, W., Meng, J., and Weaver, D., "A Fast Time-Sliced Multiple Data Bus Structure for Overlapping Data Transfers and Transformations.

4. Maples, C., Rathbun, W., Weaver, and Meng, J. "The Design of MIDAS - A Modular Interactive Data Analysis System."

5. Maples, C., Weaver, D., Rathbun, W., and Meng, J., "The Utilization of Parallel Processors in a Data Analysis Environment."

6. Meng, J., "Power Basic and the 9980/9981. ITMIX 1981 National Symposium, New Orleans, LA, March 8-11, 1981. LBL-12235, Lawrence Berkeley Laboratory, Berkeley, CA 94720.

7. Handy, Jim, "Embedded Diagnostics Utilize Excess ROM Capacity. Computer Design, May 1981, pp. 114-115.