Title
ICAMS: A NEW SYSTEM FOR AUTOMATED EMULSION DATA ACQUISITION AND ANALYSIS

Author
Arthur, A.A.

Publication Date
1983-05-01

Lawrence Berkeley Laboratory

UNIVERSITY OF CALIFORNIA

Presented at the Conference on Real-Time Computer
Application in Nuclear and Particle Physics,
Lawrence Berkeley Laboratory, Berkeley, CA,
May 15-19, 1983

ICAMS: A NEW SYSTEM FOR AUTOMATED EMULSION DATA ACQUISITION AND ANALYSIS

A.A. Arthur, W.L. Brown, Jr., E.M. Friedlander,
H.H. Heckman, R.W. Jones, Y.J. Karant,
and A.D. Turney

May 1983

NUCLEAR SCIENCE DIVISION

Prepared for the U.S. Department of Energy under Contract DE-AC03-76SF00098

ICAMS: A NEW SYSTEM FOR AUTOMATED EMULSION DATA ACQUISITION AND ANALYSIS

Allan A. Arthur, William L. Brown, Jr., Erwin M. Friedlander, Harry H. Heckman, Ronald W. Jones, Yasha J. Karant, and Arthur D. Turney
Nuclear Science Division, Lawrence Berkeley Laboratory
University of California, Berkeley, CA 94720

ICAMS (Interactive Computer Assisted Measurement System) is designed to permit the acquisition and analysis of emulsion scan and measurement data at a rate much faster than any existing manual techniques. It accomplishes this by taking the burden of stage motion control and data recording away from the scanner and putting it onto the computer. It is a modern distributed network system, where a central PDP-11 computer running under RSX-11M V4 communicates with two-ported UNIBUS memory; each two-ported memory resides on the local intelligence of each independent ODS (Optical Data Station). The intelligence of each ODS is a 6512 microcomputer running under FORTH and enhanced with a floating point processor card. Each ODS is supported on ICAMS using the virtual memory features of FORTH, permitting full access to the disk storage facilities of the PDP system. To the scanner, each ODS is conversational and menu driven. To the physicist, utilities have been written that permit FORTRAN-77 programs to easily acquire and thus analyze the data base on the PDP-11.

A general difficulty with visual techniques in high energy physics has been the relatively slow data-taking rate, caused by the labor-intensive nature of the experiments, and the resulting low statistics of the final data base. On the other hand, visual techniques, and especially research emulsion, have the intrinsic advantages of essentially complete 4π coverage of all charged particles from the target-, mid-, and projectile-rapidity regions. Emulsion has excellent charge resolution, since in one target-detector charges of minimum ionizing tracks from charge 1 to charge 92 can be determined. With these intrinsic advantages of emulsion, it seemed reasonable to use modern data acquisition and reduction techniques to relieve the labor-intensive functions.

ICAMS (Interactive Computer Assisted Measurement System) is our answer to this. It is a distributed network system. The system has two major components, the central computer and individual data-taking stations, called ODS (Optical Data Station). The central computer is a Digital Equipment Corporation PDP 11/23 with a 22-bit address space (of which 1 Mbyte is currently implemented) running under RSX-11M V4, with floating point, eight asynchronous I/O lines (one line printer, one system console, one dial-up port, and five CRT terminal ports), four RL-02 disks, and an Ampex 330 Mbyte Capricorn Winchester disk emulating an RM-05. All the preceding hardware resides on the LSI-11 bus. However, we do need access to UNIBUS peripherals. For this aim, we have an ABLE UNIMAP, which provides a conversion between the UNIBUS and LSI-11 bus, and also provides the UNIBUS mapping registers present on a PDP 11/24, 11/44, or 11/70. The system also has a TMA-11/TU-10 800 BPI 9-track tape drive, which is a UNIBUS peripheral. More importantly, the network actually communicates via the UNIBUS.

Each ODS is equipped with a 6512 microprocessor using the Motorola bus and is also equipped with a Creative Micro Systems 9611 arithmetic processor (32 bit floating point unit with higher "slide rule" functions). In addition, a special LBL-designed two-port memory is included, which appears as 512 bytes addressable read-write memory on both the UNIBUS and the Motorola bus. Thus, UNIBUS protocol and arbitration implements the network hardware standard on ICAMS.

The microscope stage is fitted in both axes with incremental encoders and DC servo motors on one millimeter pitch lead screws. The encoders are 1000 count per revolution, thus providing one micrometer resolution. The encoder outputs control 24 bit up-down counters, allowing the zero reference for both X and Y axis to be set to any point within the limits of stage travel. The counters are microprocessor read and are buffered such that changes during the three byte reads are inhibited. The servo amplifiers are digital phase locked loops where the reference frequencies are obtained from 15 bit rate multipliers. The master rate is obtained from the crystal controlled system clock. Either axis rate multiplier source may be from the master or slaved from the other axis rate multiplier output. Thus the stage may be driven at any angle and at any desired speed up to a maximum of 3.2 centimeters per second.

A track ball is provided; its output is used by the microprocessor to either modify a computer-directed movement or allow direct position control by the operator. When being used for direct position control the response is nonlinear to provide both accurate position control and rapid motion when moving substantial distances.

Limit switches directly prevent further stage motion and are read by the microprocessor for program modification as desired.

The software implementing ICAMS has several constraints. It must be user friendly to both the end-level scientific user, principally programming for data analysis in standard PDP FORTRAN-77, and forgiving to a scanner working on an ODS. In fact, to the scanner, the ODS should appear much as a constant companion and electronic notebook, reminding the scanner of the current scanning instructions for the experiment in progress and likewise automatically recording the data for later analysis by the physicists. This last step greatly increases the data throughput and reliability, for the previous method of hand-recording data and then entering the data to a computer by either card punch or interactive CRT terminal programs is fully eliminated. Additionally, full advantage must be made of the powered stage control of each ODS, removing the physical tedium of manually turning X and Y crank wheels to follow a track or to return to a specified point in the plate.

To implement this, it was decided to put each two-ported memory at the same address on the Motorola bus and on different but contiguous addresses on the UNIBUS. Hence, each ODS could be identified at least by its address on the UNIBUS. Since the two-port memory does not have hardware interrupt or DMA capability, rather than construct a driver in the strict RSX sense of the term, an installed polling task under RSX would be sufficient. In fact, it is possible to write such a task using FORTRAN-77, in which the contiguous ODS two-port memory is an installed partition identified to the handler task as a labeled FORTRAN common. By using appropriate handshaking and command protocols, it is possible to implement FILES-11 structures on the ODS.

Because of the time involved in programming the ODS 6512 in direct assembly language, some other higher level language was desired for general scientific scanner program implementation. On the other hand, the execution speed of assembler programs was desired to implement the direct stage control motion functions. For this reason, it was decided to have two processes time share the 6512 resources; these are the stage control process (written in assembler and fixed in EPROM), and the scientific control process, written in a higher level language. Since FORTRAN is not readily available for the 6512 (Apple FORTRAN is in principle available but too cumbersome for many of the applications desired), we selected AIM-65 FORTH, available from Rockwell as a PROM set. This language interfaces directly with assembler routines, has its own structured version of the AIM-65 assembler, and permits high level structures to be implemented, while still executing rapidly enough for real time applications. It is modular and structured; has preexisting 8, 16, and 32 bit word lengths as needed; and has two additional virtues. It is a virtual operating system (unlike RSX but akin to VMS in this respect), and it is extensible. That is, FORTH permits the compiler to be modified from within the compiler to introduce new constructs via the <BUILDS and DOES> words. Unfortunately, the intrinsic FORTH system only addresses two bytes of disk sector numbers, whereas FILES-11 under RSX addresses four bytes of disk sectors. However, we modified the FORTH PROMs to address the full four bytes implemented under RSX.

As an example of the mixed implementation, we invented the FORTH word MOUNT. When invoked by an ODS, the ODS handler task on the PDP-11 executes a FORTRAN OPEN statement, the file being opened on FORTRAN logical unit equivalent to the ODS number and the file name being what the ODS requested. Thus, in FORTH, MOUNT DL2:TESTIT.FTH. Here, DL1:TESTIT.FTH is an ASCII string transmitted via the two-port memory from the ODS to the PDP. The equivalent to the CLOSE instruction is DISMOUNT. Several file types have been added to the RSX standard file types. These are .FTH for FORTH source files (created on the PDP by editor programs written in MCR Indirect Command File language and FORTRAN-77, as the intrinsic FORTH or AIM editors are cumbersome for this application), .LOD for load files, .HED for header files, and .ODA for ODS data files.

A load file contains executable FORTH instructions, which do nothing more than cause FORTH source files to be mounted and compiled and, ultimately, some appropriate scanning program word to be executed. A header file contains FORTH source to be compiled containing the structural information for the appropriate ODS data file, a "format" of sorts. The ODS data file obviously contains the actual data of the experiment.

To keep the various pointers to a .ODA file separate, we used the intrinsic FORTH concept of VOCABULARY. However, we used the intrinsic extensibility of the compiler to force it to compile during execution from ASCII text strings created under program control. This is roughly equivalent to causing a FORTRAN program to first invoke an editor to create a subroutine, then have the same FORTRAN program invoke the FORTRAN compiler to compile the source, invoke the linking loader to actually append the object modules to the task image, and finally call the subroutine. While this is nontrivial in most FORTRAN implementations, it is readily accomplished in FORTH.

In practice, then, the software proceeds through the following steps. A master program is loaded and executed and left in charge of the ODS at all times. This program is a menu that provides the scanner with the approved choices of use of the ODS that are desired by the physicists. By selecting a menu choice, the ODS requests a .LOD file from RSX. This .LOD file is then compiled and executed; it lists the actual .FTH files that contain the program to be executed based on the scanner's menu choice.

Each .FTH file is requested from RSX and compiled into the ODS. The final command on the .LOD file is to actually execute a word (program) from one of the .FTH files (generally, the last one loaded); this begins actual use of the ODS by the scanner. By specifying a particular experiment and plate, the physics program specifies a .ODA file, which is requested from RSX and created if it does not already exist (the OPEN statement in the ODS polling task on ICAMS does this). The first part of the .ODA file name indicates what it is; e.g., AN3261.ODA refers to an ANgle measurement file in emulsion stack 326 (^{3}Fe is the actual stack name, but Fe has a charge of 26, of course), plate number 1, and likewise a new VOCABULARY is automatically created. The VOCABULARY name is just the first part of the file name, e.g., AN3261.ODA. The header file is then requested from RSX under extension .HED; for all angle measurement data bases, the file name is AN, e.g., AN.HED. This file is loaded under the VOCABULARY so that data can be uniquely stored on the RSX controlled disk. For example, if X is the generic name of an x coordinate and two separate plates are under measurement (say plate 1 in stack 326, VOCABULARY name AN3261, and plate 5 in stack 326, VOCABULARY name AN3265), the pointer to the first data would be the FORTH fragment AN3261 X, while the second would be AN3265 X.

Actually, things are a bit more complicated than this, because the coordinate information is either kept as a virtual array (the same as an array in physical memory but actually residing on the RSX controlled disk), or as a linked list, so some information must be given before the call to X. This information plus the word X under the appropriate VOCABULARY generates an address to the virtual operating system of FORTH and permits data to be fetched or stored as needed.

The data are then available for analysis by the physicists using FORTRAN-77 programs under RSX, using the non-ODS portion of ICAMS, which also serves as a general purpose computer, text editor, etc. Special data acquisition subroutines are written, of course. These FORTRAN-77 compatible subroutines first parse the appropriate .HED file to determine the pointers into the data file on the disk and then read the data as requested by the physicist-supplied data analysis program.

This work was supported by the Director, Office of Energy Research, Division of Nuclear Physics of the Office of High Energy and Nuclear Physics of the U.S. Department of Energy under Contract DE-AC03-76SF00098.

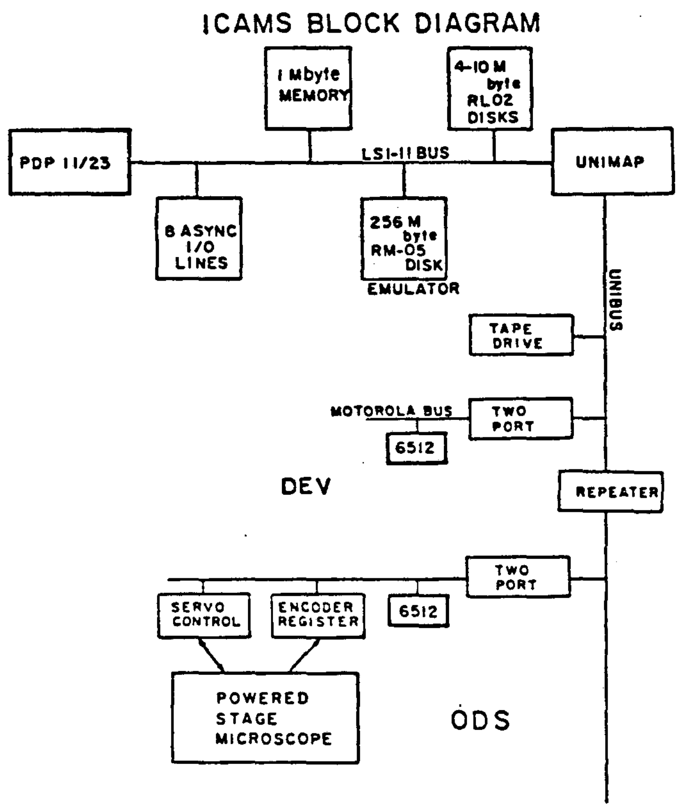
ICAMS BLOCK DIAGRAM
I Mbyte MEMORY
4-10 M byte RL02 DISKS
PDP 11/23
LSI-11 BUS
UNIMAP
8 ASYNC I/O LINES
256 M byte RM-05 DISK EMULATOR
UNIBUS
TAPE DRIVE
MOTOROLA BUS
TWO PORT
6512
DEV
REPEATER
TWO PORT
SERVO CONTROL
ENCODER REGISTER
6512
POWERED STAGE MICROSCOPE
ODS

Fig. 1

TECHNICAL INFORMATION DEPARTMENT
LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720