LA-5079-MS

INFORMAL REPORT

MASTER

A Proposal for EPICS Data Acquisition Software

los alamos
scientific laboratory
of the University of California
LOS ALAMOS, NEW MEXICO 87544

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

In the interest of prompt distribution, this LAMS report was not edited by the Technical Information staff.

Printed in the United States of America. Available from
National Technical Information Service
U. S. Department of Commerce
5285 Port Royal Road
Springfield, Virginia 22151
Price: Printed Copy \$3.00; Microfiche \$0.95

LA-5079-MS

Informal Report

UC-28 & 34

ISSUED: November 1972

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

A Proposal for EPICS Data Acquisition Software

by

R. A. Eisenstein*

P. D. Barnes*

H. A. Thiessen

*Guest scientists. Permanent address: Carnegie-Mellon University,
5000 Forbes Avenue, Pittsburgh, PA 15213

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

A PROPOSAL FOR EPICS DATA ACQUISITION SOFTWARE

by

R. A. Eisenstein, P. D. Barnes, and
H. A. Thiessen

I. INTRODUCTION

The EPICS Channel at LAMPF is designed to collect high resolution data from reactions like (π, π) and (π, π') . Incident pion energies range from 50 to 300 MeV; available scattering angles are from $\theta \sim 20^\circ$ to $\theta \sim 150^\circ$. For the purposes of discussion, we assume that the standard data collection apparatus is configured as schematically shown below. (Documents containing information regarding the magnetic optics and the electronics system are given in Ref. 1-7.)

T = Quadrupole Triplet

S = Spectrometer

V1-V4 = Pole Tip Veto Counters

HC1-HC4 = Helical Proportional Chambers

S1, S2 = Trigger Scintillators

LC = Lucite Cherenkov Counter

GC = Gas Cherenkov Counter

Particles travelling through all or part of the above system are subject to a series of fast electronic tests⁴ which determine whether the event is suitable for further processing via the computer-based data acquisition system. A good event is

strobed into the CAMAC (Microprogrammed) Branch Driver (MBD);⁵ then directly into the memory of a PDP-11/45 computer. A good event consists of some 34 16-bit words within the MBD (see Appendix A). Since many of these words require only one byte instead of two, it is possible to repack the event (via MBD) into approximately 22 PDP-11 16-bit words. (Appendix A.) The packed output from the MBD shown in Appendix A is at the highest useful density. The MBD has been used to pack words and also to convert some of the raw data into derived quantities (e.g., x-coordinate calculated from x-coarse and stops A,B,C). The price of the higher packing density is thus a loss of information (e.g., possibility of measuring the time jitter in the stop A,B,C TDC outputs). The saving in recording magnetic tape space that packing allows may not justify this loss of information. Hence, the best role for the MBD in our data acquisition scheme has not been decided.

It is clear from the above that even moderate event rates imply large word rates into the computer. The expected event rates at EPICS are governed primarily by the duty factor of the accelerator and the time required to prepare an event for entrance into CAMAC:

$$\text{Event Rate} = \frac{(60,000 \text{ } \mu\text{s/sec})}{50 \text{ } \mu\text{s/event}} = 1200 \text{ events/sec}$$

$$\approx 24\text{K words/sec.}$$

The rate will be slower than this because of the additional time required to bring the event from the MBD into the PDP-11. If the MBD must request the bus for every word transferred, then the time required to gain control of the bus adds a large overhead:

$(3.5 \mu\text{s Bus grant time}) \times (\# \text{ of words}).$

This is between 80 and 120 μs , depending on the packing scheme used. If it can be arranged so that the MBD must acquire the bus only once, during which it transfers all its words, then the additional time required is about 35 μs . Total event time is thus

$$\begin{array}{rcl} 35 \mu\text{s} & 85 \mu\text{s (one unibus request)} & \\ 50 \mu\text{s Electronics time + or} & = & \\ 80-120 \mu\text{s} & \sim 150 \mu\text{s (unibus request for each word)} & \end{array}$$

There are about 120 machine pulses/sec, each 500 μs long. Hence there will be roughly 8 millisecc between bursts, during which between 3 and 6 events need be processed by the computer if it is to stay even with the data rate. The goal of the data acquisition software is obviously to make most efficient use of the time available logging and analyzing data and monitoring the experiment.

The above paragraph suggests that a multilevel priority scheme be developed under which the software operates.⁷ A broad outline of the priority scheme that has been adapted follows:

Highest Priority

1. Input Data
2. Validity Checks on Data (Data Filter)
3. Log valid data on magnetic tape
4. Provide experiment monitoring (histograms, instrument control, displays, etc.)
5. Secondary Data Analysis, Histogramming, Display
6. Creation of "Good Event" Files on a secondary mass storage device
7. Further Processing and Data Analysis.

At low data rates, the normal mode of the software is in the lower priority regions. As rates increase, less time is available for these lesser tasks; at the highest rates, secondary analysis is not possible. All priorities are allocated dynamically and are driven by interrupts external to the computer.

A primary consideration in the design of this software has been the need for a very flexible and general data acquisition system. In particular the package should be easy to initialize and run; it should be possible to modify it easily so that a minimum of data collection time is lost if changes

are necessary. Examples of such modifications are

1. A re-ordering of the priorities listed above
2. Changes in the Data Validity Checks
3. Changes in Histograms and Display
4. Changes in Instrumental Control Values.

An important job for this program will be in the analysis of data tapes after runs have been completed.

Hence, it must accept and analyse data from several possible sources with equanimity.

To effect these goals, the software has been designed in a modular form. We have sought to employ fully the advantages of the PDP-11 system in our use of the interrupt structure and the capabilities for writing re-entrant subroutines. This allows considerable saving of memory space, as routines used in several places may be interrupted, re-entered, and nested to any depth. The small overhead required to run a program of this type is more than repaid by the increased flexibility.

Finally, we seek a data acquisition system that is compatible with the needs of other users. It is conceivable that a laboratory-wide data acquisition package for PDP-11's can be developed; the manpower saved in development and maintenance of such a package would be very large. At a minimum, we urge the development of the following lab standard software:

- 1a. Generalized Data Testing Procedures
- 1b. Histogram Packages for core-resident and disk-resident histograms
2. Display Package for Tektronix 4010
3. Tektronix 4010 Keyboard device driver
4. Real-time Clock Device Driver.

Suggestions for a way of implementing 1a. and parts of 1b. are given below. Initial attempts at coding indicate that it is possible to create a system with the above-mentioned features that is also efficient and fast.

II. PROPOSED EPICS DATA FLOW

A Data Flow Scheme for EPICS is shown in Fig.

1. Conceptually, the EPICS data acquisition problem

Fig. 1. Proposed Data Flow Scheme for EPICS.
General scheme.

has been divided into three segments:

1. Data Source and Data Input Buffers
2. Data Transfer, or Filter Segment
3. Data Logging and Good Event Logging.

The Data Source may be either an MBD with real-time data, or a magnetic tape containing the results of past runs. The Data Source is double buffered. (Data Input Buffer.)

The Data Transfer section is a data processor. In its least sophisticated form, it performs only a unity transformation on an event, passing it directly to the logging buffers. At a higher level, it will perform numerous tasks (checks, calculations) on the data according to a priority schedule designed by the experimenter at run time. The Transfer section

shown in Fig. 2 is divided into three levels separated by a priority adjustment (PA) scheme. All levels T_A , T_B , T_C have available essentially the same processing capability and are separated only by priority. Their contents are discussed below. What is done in each level is under the experimenter's control.

Priorities can be adjusted in the following way, using T_A and T_B as examples:

1. $P(B) = P(A)$ for all events. Each event will proceed through both T_A and T_B where it will be tested, histogrammed, etc.
2. $P(B) = P(A)$ either on a conditional basis or on a numerical basis. (e.g., Put 1 event in every 10 through both T_A and T_B ;

Fig. 2. Data Transfer Section Schematic.

or, put an event through T_B only if it passes some test in T_A .)

3. $P(B) < P(A)$, and therefore events are processed by T_B only on an available time basis.

Similar statement applies when T_C is included. In the scheme shown in Fig. 2, the logging tape is shown connected to level T_A , but by a suitable adjustment of priorities, T_B and T_C can be made a part of T_A . Level priorities are assigned at run time and provisions will be made to be able to change them while a run is in progress. Input data and program control are discussed later.

The common areas CA, CB, CC are used as temporary single event storage for T_A , T_B , and T_C . In this way, events being analyzed remain unaltered at a given level should an interrupt occur.

The Data Transfer diagram has included a provision for a second tape unit, the Good Event log tape. If desired, the experimenter will be able to more completely analyze events and store results on this second tape. Hopefully this will reduce the amount of analysis required off-line. He can, for example, choose to analyze every event "completely" reducing later work significantly.

A flow chart providing for a system as described above is given in Appendix B.

III. TRANSFER CODES

The transfer codes T_A , T_B , T_C are in the routines responsible for data filtering. A maximum of three software levels of priority is thus possible; the minimum number is one. In addition, each level may range in complication from performing no operation on the data to performing sophisticated data reduction and management tasks. A flexible scheme has been devised which allows easy and complete specification of all transfer codes at set-up time, as well as easy modifications later. The method used is outlined below.

A block diagram of a section of transfer code is shown in Fig. 3. (A slightly more detailed version is given in Appendix B with the remaining flow charts for the entire system.) Each block shown in the figure is essentially a single subroutine. All of these are re-entrant and need only exist once in core. Each is called via the sequence

JSR Register, SUBR

--

Arg 1

Arg 2

.

.

Arg N

The arguments listed below the call are the arguments required for operation of the subroutine. When the program is compiled, the subroutine calls with "Standard" arguments are arranged in the order shown previously, and the correct arguments are inserted at run time by means of the Input routine or by previously executed code. (More information on this appears in the Input and Control section.) With only slight modification of this idea, the subroutines can be made to execute in any order, more than once, or conditionally, even though the calls are compiled in a fixed order. (See Appendix C.) This allows great flexibility in the operation of this software.

For example, an event may be rejected after only one test, or at any point along a series of tests, by simply calling the MICROTTEST or MACROTTEST routines several times, each time only doing one (or a few) tests. It is also easily possible to provide for more than one USER calculation per level. These points are discussed in more detail below.

In the ordering shown previously for T_A (the other transfer codes are similar), the first task is to move the raw data from one event into common area CA. Next, a user-supplied subroutine is called that performs one or more specific calculations using raw data. (This subroutine is linked to the program at run time via .GLOBS that provide the correct linkages.) The results of these calculations are appended to CA at the end of the raw data list. (Beginning at location CALRES.) Results may be in fixed or floating point. Next are two logical routines that provide the user with the capability to check his data and calculational results in a very general way. These are described below in more detail.

It has been concluded that any test on data that may be required can be made up of combinations of the following two MICROTTESTS:

1. LIM (N,X). Are the contents of "Item" N greater than the number X? If YES (true) set the corresponding test result bit = 1. If NO (false) set the test result bit = 0.

The "Item" may be a BYTE, a WORD or a FLOATING POINT QUANTITY.

"N" is the address of the "Item" relative to address CA.

The test can be done on raw data or on calculated results.

2. BIT (N,M). Is Bit M of word N set? If YES set test result bit = 1; if NO set bit = 0.

"N" is the address of the word relative to CA.

A list of MICROTTESTS to be performed in T_A may be specified at run time. The tests are performed in sequence, and their results (result bit = 1, or 0) are stored in the MICROTEST RESULT WORDS, beginning in location MITRW. Each such test result word can hold the results of up to 16 MICROTTESTS. An arbitrary number of MICROTTESTS may be performed; as

Fig. 3. Transfer Code Block Diagram - Level A.

many test result words as needed will be appended to CA. In order to investigate the coding problems and operating times of this crucial subroutine, a portion of it was designed and written. (BIT test and all LIM tests except those for bytes and floating point numbers.) The operating scheme and code are given in Appendix D. About 40 lines of code are required to write a re-entrant routine that will perform an arbitrary number of bit and limit tests and create the MICROTTEST result words. The estimated time required for 32 tests (1/2 passed 1/2 failed), including all overhead is

PDF-11/45	MOS Memory	450 μ s
PDP-11/20	Core Memory	1350 μ s.

Minimum overhead is about 20 (60) μ s, respectively, for no tests. The number of each kind of test, the relative addresses of the event words on which they are to be done, and the correct bit or limit for each test, are input at run time.

The MACROTTEST subroutine tests for the successful completion of a logical combination of MICROTESTS. The available logic is:

1. TRUE: [Listing by number of the MICROTESTS which must be simultaneously true (Test Bits set)]
2. FALSE: (Listing of MICROS which must be simultaneously false)
3. OR: (Listing of MICROS to be Inclusively OR'ed; at least one MICRO must be satisfied.)
4. XOR: (Listing of MICROS to be Exclusively OR'ed; only one MICRO must be satisfied.)

A given set of MICROS must pass all four component parts of a MACRO to avoid failing the MACRO. When this occurs, a bit corresponding to the number of the MACROTTEST will be set in the MACROTTEST RESULT WORD at location MATRW. It seems unlikely that more than 16 MACROTTESTS will be required at any level T, and so only one MACROTTEST RESULT WORD is used. A possible scheme for a MACROTTEST subroutine has been included in Appendix E. The required MACROTTEST combinations are entered at run time; each MACROTTEST includes entries under each of T, F, O, X.

Example

Suppose we want to "pass" all events that satisfy

$$90 \leq \text{ADC1} \leq 120$$

$$100 \leq \text{ADC2} \leq 130$$

and also (but as a new category) those satisfying the above and in addition

$$\text{ADC1} + \text{ADC2} \leq 240.$$

MICROTTESTS would be

1. LIM (1,90)
2. LIM (1,120)
3. LIM (2,100)
4. LIM (2,130)
5. LIM (3,240).

The MACROTTESTS would be

1. T: 1,3
F: 2,4
O: (none)
X: (none)
2. T: 1,3
F: 2,4,5
O: (none)
X: (none).

Obviously, no BIT MICROTTESTS are required, nor are OR and XOR MACROTTESTS.

In passing, note that we will possibly have need for tests like (for example)

$$O: (2,4,6), (8,10)$$

[Interpreted as $((2V4)V6) \wedge (8V10)$]

and also

$$X: (3,5,7), (9,11)$$

[Interpreted as $((3W5)W7) \wedge (9W11)$].

The routine in Appendix E will not handle tests of this kind.

The alternative approach to a scheme of this kind is the construction of a rigid set of in-line routines to perform the desired tests. If many tests are required, then the in-line routines become repetitive and use up core; in addition, modification of the program requires recompiling and may require significant program changes. These disadvantages might be worth their cost if the code were

significantly faster or more efficient. This does not appear to be the case, except when testing is minimal, particularly if one writes the in-line tests as subroutines, or if one wishes to store the results of his testing to avoid repetitive testing later. Then the two methods are almost equivalent in speed and efficiency. If the program is in Fortran, any possible gain in efficiency or speed will be lost in Fortran overhead. It is true that a dedicated program could probably be written that will execute faster than the equivalent MICRO-MACRO program, but unless it is much faster it would seem that the generality and flexibility of the scheme described above are much more desirable. See Appendix F.

After the MICROTTESTS and MACROTTESTS are completed, decisions must be made regarding data logging and histogramming. These decisions are based on the results of MACROTTESTS and are quite straightforward. The LOG subroutine simply asks if the event has passed a user-selected MACROTTEST, and if so it transfers the event (with calculations and all Test Result Words) into the tape output buffer area. Full buffers are logged on a demand basis by the tape handling routines.

The HISTOGRAM subroutine is slightly more complicated because it requires additional information. Our requirement for a histogrammable event is that it pass a selected MACROTTEST; the event is then converted into a histogram coordinate using information in a HISTOGRAM DEFINITION TABLE. If the histogram is resident in core the appropriate location is incremented directly; if on disk the coordinate of the word to be incremented is stored in a named table until histogram update is possible.

For some high-count-rate applications, the 16-bit word length of the PDP-11 may not be sufficient for count storage (~ 64K counts are possible) and overflows will occur. One byte overflow capability will be provided, but where they will be stored and how they will be referenced has not been decided.

It is hoped that a laboratory standard histogram (HSTORE) package can be developed which given

1. HISTOGRAM name
- 2a. Relative coordinate of word to be incremented

or

- 2b. List of relative coordinates of words to be incremented

will manage histogram updating (in core or on disk) and overflows. Appendix G contains a schematic outline of the HISTOGRAM routine and the interface to the routine HSTORE.

Another routine that is closely connected with the histogram package, but not in general a part of any level T, is the DISPLAY package. This program should also be a laboratory standard routine that when called as a subroutine with an appropriate argument list, will display any section of a histogram (up to three dimensions) using the Tektronix 4010. A lab standard package has not yet been defined.

A Dot Plot capability should be included in this software. A dot plot is understood to mean a large number of (x,y) pairs, each plotted on the 4010 as a dot when they enter the computer. Because of core limitations, no other record or list of them is kept. A 4010 scope camera will allow one to keep a permanent record of these plots.

The last subroutine to be considered in this section is the PRIORITY ADJUST subroutine. Although full details have not been decided yet, the general scheme will involve a set of software switches (or flags) that will govern the future course of the program, as outlined in Section II. These switches can be set at run time and modified dynamically via the keyboard of the 4010.

IV. INPUT AND CONTROL

Input to the program at Run Time will be via an interactive program using the 4010 display and keyboard. The program will contain preset (standard) values for most quantities which may be changed on a demand basis by the user. Input information will be grouped as follows:

At Load Time:	1. Link to subroutine that Loads MBD Code 2. Links to subroutines for <u>user</u> calculations at levels T_A, T_B, T_C.
At Run Time:	1. General Run Information (date, time, experiment, etc.) 2. Priorities of levels A,B,C ($P(A) \geq P(B) \geq P(C)$) 3. Level A - Definitions for: a) Microtests

- b) Macrotests
- c) Logging Information
- d) Histogram Information
- 4. Level B (as in (3))
- 5. Level C (as in (3))
- 6. Display Information
- 7. Magnet and other control information for experimental monitoring and control.

The run is then started. The program proceeds operating along the priority lines assigned by the user. At present intervals, the program will be interrupted by the real time clock for the following tasks:

- 1. Display update (once every 10 seconds?)
- 2. Magnet and other control monitoring (once in 5 minutes?)
- 3. Other ?

The program will also be interrupted on a random basis by the following devices:

- 1. The MBD on Word Count Overflow
- 2. The magnetic tape units, on Word Count Overflow
- 3. Keyboard Interrupts from the 4010.

The last type of interrupt will ordinarily be from a user wanting some type of service on a dynamic basis. This might require

- 1. Change in a current display
- 2. Change to a new display
- 3. Output to the display of current run status information (magnet settings, control settings, etc.)
- 4. Terminating a run
- 5. Restart of all or part of the Input program to re-initialize the run
- 6. Change in MICRO, MACRO or HISTOGRAM definitions at a given level
- 7. Change in the order of execution of the subroutines at a given level
- 8. etc.

To service this type of request for service, a keyboard interpreter will be needed for the 4010. Presumably, the present control typewriter device driver can be used for part of this interpreter. In passing, we note that not all of the functions listed above are simple enough to be done unambiguously as dynamic changes in an executing program. They will instead require the orderly termination

of an existing run, with careful records kept of changes made, and a re-initialization of the program.

One of the authors (RAE) wishes to acknowledge very useful extended conversations with his colleagues J. Amann, Jennie Boring, M. Kellogg, Max Seamons, and R. Thomas. The valuable comments of D. Gemmell, R. Mischke, and B. Preedom have shown alternative viewpoints and possible shortcomings in the present system. Future remarks will be most welcome.

Jim Amann is continuing work on the scheme presented here. Improvements have been made with regard to MICRO and MACRO test procedures. More details will be available soon.

References

- 1. H. A. Thiessen, "A Proposal for EPICS: A High Resolution Beam and Spectrometer Facility for Nuclear Structure Research at LAMPF," LASL Report LA-4534-MS (October 1970).
- 2. H. A. Thiessen, M. Thomason, "Final Design Information, EPICS Spectrometer," (unpublished, informal document).
- 3. R. A. Eisenstein, P. D. Barnes, "Epics Data Acquisition System (Spectrometer Optics," Internal Report MP-7-63 (LAMPF-MP-7).
- 4. H. A. Thiessen, "Proposed Fast Electronics for EPICS," LASL Report MP-7-60 (to be issued as LAMS report).
- 5. D. M. Lee, S. Sobottka, H. A. Thiessen, "A Bifilar Helical Multiwire Proportional Chamber for Position Sensitive Detection of Minimum Ionizing Particles," LASL Report LA-4968-MS (August 1972).
- 6. L. R. Biswell, R. E. Rajala, "A Microprogrammed Branch Driver (MBD) for a PDP-11 Computer," LASL Report LA-4916-MS (informal Report UC-32).
- 7. R. A. Eisenstein, "Considerations Concerning a Computer-Based Data Acquisition System for EPICS," Internal Report MP-7-63 (LAMPF-MP-7).

APPENDIX A

ODD BYTE		EVEN BYTE	
C212		C212	
		C212	
F	X1 width	X1-coarse	
F	Y1 width	Y1-coarse	
F	X2 width	X2-coarse	
F	Y2 width	Y2-coarse	
F	X3 width	X3-coarse	
F	Y3 width	Y3-coarse	
F	X4 width	Y4-coarse	
F	Y4 width	Y4-coarse	
LRS 2226A QUAD TDC		0 H	ELICAL 1
		0 H	ELICAL 2
		0 H	ELICAL 3
		0 H	ELICAL 4
QUAD TDC		0 S	TOP A-1
		0 S	TOP B-1
		0 S	TOP C-1
		0 S	TOP A-2
QUAD TDC		0 S	TOP B-2
		0 S	TOP C-2
		0 S	TOP A-3
		0 S	TOP B-3
QUAD TDC		0 S	TOP C-3
		0 S	TOP A-4
		0 S	TOP B-4
		0 S	TOP C-4
LRS 2248 OCTAL ADC			VETO 1
			VETO 2
			VETO 3
			VETO 4
			S1
			S2
			GAS CHER.
			OPTIONAL

INPUT TO MBD

34 WORDS

F = FAULT

0 = OVERFLOW

☐ = 2 BITS

ODD		EVEN	
C212		C212	
--		C212	
V1		V2	
V3		V4	
S1		S2	
GC		OPTIONAL	
-	0	H	ELICAL 1
-	0	H	ELICAL 2
-	0	H	ELICAL 3
-	0	H	ELICAL 4
F	-	x width	X-COARSE
F	-	y width	Y-POSITION
X-POSITION		X-POSITION	
F	-	x width	X-COARSE
F	-	y width	Y-POSITION
X-POSITION		X-POSITION	
F	-	x width	X-COARSE
F	-	y width	Y-POSITION
X-POSITION		X-POSITION	
F	-	x width	X-COARSE
F	-	y width	Y-POSITION
X-POSITION		X-POSITION	

POSSIBLE MAXIMALLY PACKED

OUTPUT FROM MBD

22 WORDS

NOTES

- 1) OVERFLOW ON 2248 GIVES ALL BITS SET
- 2) NORMAL OUTPUT OF 2226A CAN HAVE 9 BITS. 10th BIT SET ON OVERFLOW.
- 3) ASSUME HC Y-COORDINATE REQUIRES ONLY UP TO 8 BITS. X-COORDINATE REQUIRES 12-14 BITS. FAULT AND WIDTH BITS RETAINED; STOPS A, B, C ARE LOST IN CONVERSION TO COORDINATE X.

APPENDIX B
INTERRUPT PROCEDURE, (DIB Section)

- + EVENT IN COMPUTER MEMORY VIA DMA
- * TO NEW ADDRESS IN CURRENT DIB
- ** TO TOP ADDRESS IN ALTERNATE DIB
- *** THIS POINT COULD ALSO SERVE AS STARTING POINT FOR THIS SECTION OF CODE.

NOTES

- 1) $P(MBD) < P(MT) = P(DISK)$
- 2) ASSUME MBD INPUTS 1 EVENT AT A TIME.

WCO = WORD COUNT OVERFLOW
 WC = WORD COUNT
 CA = CURRENT ADDRESS

T_A Scheme

T_B Scheme

TAPE BUFFER CONTROL

FINISH FLAG = 1 IS NORMAL STATE ONE BUF IN
PROCESS OF TRAN. TO TAPE OTHER
BUF. FILLING.
FIN. FL. = 0 IS FREE STATE FILLED BUF. TRANS.
COMPLETE OTHER BUF. FILLING
FIN. FL = 2 IS FULL STATE FILLED BUF. TRANS.
INCOMPLETE OTHER BUF. FULL.

APPENDIX C

Subroutines, even though ordered as shown at Compile time, may be executed in any order using the method below.

Assume the order of execution of the subroutines is not as shown. To do this, a table is prepared each entry of which is the address of the call that is desired. For example,

MOV
CAL
MOV
MIT
MAT
LOG
CAL
MOV
:
PRI

Let R5 be the table pointer. Then the following code will execute the routines in the desired order.

```
LIST TST @ (R5) +
      BEQ LIST
      JMP @ (R5)
```

Deletions are possible by zeroing a table entry. Last routine called must be PRIOR.

Location

MOV	JSR R4, <u>MOVDAT</u> (ARGUMENTS) JMP LIST
CAL	JSR R4, <u>ACALC</u> (ARGS) JMP LIST
MIT	JSR R4, <u>MITST</u> (ARGS) JMP LIST
MAT	JSR R4, <u>MATST</u> (ARGS) JMP LIST
HIST	JSR R4, <u>HISTO</u> (ARGS) JMP LIST
LOG	JSR R4, <u>LOGTAP</u> (ARGS) JMP LIST
PRI	JSR R4, <u>PRIOR</u> (ARGS) JMP LEVELB

STANDARD SUBROUTINE CALL LIST

Dummy JSR to other .GLOBL routines can be included in the compiled list but only called when needed from the table.

APPENDIX D

MICROTEST

These subroutines do MICROTESTS and write the Result word.

BIT-LIMIT TEST ROUTINE [DOES NOT INCLUDE BYTE OR FLOATING POINT]

Definition: Re-entrant routine for fixed point limit test, the bit test and creation of test words.

Call: J&R R5,MITST : CALL SUBR FOR BIT/LIM TEST
 BNUM : # OF BIT TESTS
 LNUM : # OF LIMIT TESTS
 MITRW : ADDR. OF 1st MICROTST RES. WORD
 WADR : LIM TST BUFFER ADD. POINTER
 BADR : BIT TST BUFFER ADD. POINTER

Subroutine:

MITST MOV R1, -(SP) : SAVE R1 ON STACK
 MOV (R5)+, -(SP) : SAVE BNUM ON STACK
 MOV (R5)+, -(SP) : SAVE LNUM ON STACK
 MOV (R5)+, -(SP) : SAVE MITRW ON STACK
 MOV TSTBIT, -(SP) : SAVE TSTBIT ON STACK
 MOV (R5)+, R1 : MOVE WADR TO R1

LTST DEC +4(SP) : CHECK # OF LIM TSTS ≥ 0
 BLT BITBEG : FINISHED? GO TO BIT TST

 CMP @ N(R1), (R1)+ : NO, DO LIM TST
NAD .-2 : LIMTST BUF. SPACE ≥ # OF TSTS
 BLOS JLU : TEST FALSE? LEAVE BIT ZERO
 BIS +2(SP), +4(SP) : TEST TRUE, SET BIT

JLU CLC : CLEAR C BIT
 ROL (SP) : ROT TSTBIT LEFT
 BCC LTST : INTO CARRY? NO, NEXT TEST
 ROL (SP) : YES, ROT AGAIN
 SUB #2, +4(SP) : AND UPDATE TEST WORD POINTER

 BR LTST : CONTINUE TESTING

BITBEG MOV (R5)+, R1 : MOVE BADR TO R1
BTST DEC +6 (SP) : CHECK # OF BIT TESTS ≥ 0
 BLT ENDBL : FINISHED? GO TO END

 BIT @ M (R1), (R1)+ : NO, BIT TEST
MAD .-2 :
 BEQ JBU : TEST FALSE? LEAVE BIT ZERO
 BIS +2 (SP), -4 (SP) : TEST TRUE, SET BIT

JBU	CLC	: CLR C BIT
	ROL (SP)	: ROT TST BIT LEFT
	BCC BTST	: INTO CARRY? NO, NEXT TST
	ROL (SP)	: YES, ROT AGAIN
	SUB #2, +2 (SP)	: AND UPDATE TST WRD PNTR
	BR BTST	
ENDBL	ADD #12, SP	: RESTORE STACK
	MOV (SP)+, R1	: RESTORE R1
	RTS R5	: RETURN
TST BIT	000001	

All integers above are octal.

APPENDIX E

Macrotest definitions are loaded at INPUT. They are contained as shown below in a Macrotest Area. Since each logical operation T, F, O, or X must be capable of testing all Microtest result bits, the number of words specified in T, for example, must be the same as the number of Microtest Result Words.

Only 16 macrotests are allowed (at present) per level.

MACROTEST

MACRO TEST ROUTINE [INCLUDES ONLY OR TESTS LIKE (4 V 3 V 5---) AND XOR TESTS LIKE (5 V 6 V 9 V---). SEE TEST]

Call	JSR RS, MATST	
+22	NMAT	: # OF MACROS TO DO
+20	NMITW	: # OF MICROTEST RESULT WORDS <u>X2</u>
+16	MITRW	: ADD. OF 1st MICROTEST RESULT WORD
+14	MATW	: ADD. OF 1st MACROTEST TEST WORD
+12	MATRW	: ADD. OF MACROTEST RESULT WORD

MATST	MOV	(R5)+, -(SP)	: # OF MACROTESTS TO DO
	MOV	(R5)+, -(SP)	: # OF MICROTEST WORDS
	MOV	(R5)+, -(SP)	: ADDRESS OF MICROTEST RES. WORD
	MOV	(R5)+, -(SP)	: ADDRESS OF MACROTESTING WORD
	MOV	(R5)+, -(SP)	: ADDRESS OF MACROTEST RES. WORD
	SUB	#12, SP	: +10 IS STROBE BIT
			: +6 IS MATW FOR UPDATING
			: +4 IS 4x(NMITW)
			: +2 IS MICROTST WORD COUNTER
	MOV	TSTBIT, +10(SP)	: 0 IS FREE WORD
	MOV	+20(SP), +4(SP)	: MAKE POINTER INCREMENT FOR
	ROL	+4(SP)	: NEXT MACROTEST (i.e., 4X(NMITW))
	ROL	+4(SP)	
	CLR	+2(SP)	: CLEAR MICRO COUNTER
TRUE	MOV	+14(SP), +6(SP)	: KEEP MATW ADDRESS SAFE
			: WILL INCR. (SP)
	TST	@ + 6(SP)	: IS MATW ZERO? IF SO, GO
	BEQ	FALSE	: ON TO FALSE TEST.
	MOV	@ + 6(SP), (SP)	: NOT ZERO, TEST AGAINST MICROTEST
	BIC	@ + 16(SP), (SP)	: RESULT
	BNE	NOGOOD	: NOT SATIS., TERMINATE MACROTEST
FALSE	ADD	+20(SP), +6(SP)	: POINT TO FALSE TST WORD
	TST	@ + 6(SP)	: IS TST WORD ZERO?
	BEQ	ORT	: YES, GO ON TO OR TST
	MOV	@ + 6(SP), (SP)	: MOVE MACROTST WD INTO FREE
	COM	@ + 16(SP)	: AREA. COMPLEMENT MICRO TEST. DO
	BID	@ + 16(SP), (SP)	: "TRUE" TST. RECOMPLEMENT.
	COM	@ + 16(SP)	: NOT SATISFIED, TERMINATE
	BNE	NOGOOD	
ORT	ADD	+26(SP), +6(SP)	: POINT TO OR TEST WORD
	TST	@ + 6(SP)	: IS TEST WORD ZERO?
	BEQ	XORT	: YES, GO TO XOR TST
	BIT	@ + 16(SP), @ + 6(SP)	: NO, AND MICRTST AND MACRO
	BEQ	NOGOOD	: NO BITS IN COMMON, FAILS
XORT	ADD	+20(SP), +6(SP)	: POINT TO XOR TEST WORD
	TST	@ + 6(SP)	: IS TEST WORD ZERO?
	BEQ	NEXT	: YES, GO TO NEXT PROCESS
	MOV	@ + 6(SP), (SP)	: NO, MOVE TEST INTO WORK AREA
	ROL	(SP)	: ROTATE LEFT (CLRS C IF 15 CLR)
	BCC	.-2	: CARRY CLR? YES, BRANCH
	TST	(SP)	: NO, TEST FOR OTHER BITS SET
	BNE	NOGOOD	: IF (SP) IS ZERO, TEST OK

NEXT	ADD	#2, +2(SP)	: CHECK IF MACRO TEST DONE
	CMP	+20(SP), +2(SP)	
	BEQ	UPDATE	: FINISHED TEST, UPDATE MATRW
	ADD	+4(SP), +14(SP)	: NOT DONE, CHANGE MATW
	BR	TRUE	: GO ON
UPDATE	BIS	(SP), @ +12(SP)	: UPDATE MATRW
	CLC		: CLEAR C
	ROL	(SP)	: ROTATE TEST BIT
	BCS	MOPUP	: NO MORE THAN 16 TESTS
	BR	MORTST	: MORE MACRO TESTD?
NOGOOD	CLC	(SP)	: ROTATE TEST BIT
	BCS	MOPUP	: NO MORE THAN 16 TESTS
MORTST	DEC	+22(SP)	: DECREMENT MACTST COUNTER
	BEQ	MOPUP	: IF ZERO, STOP
	CLR	+2(SP)	: NOT ZERO, CLR MICTST WORD CNTR
	ADD	+4(SP), +14(SP)	: NEW POINTER TO MAC TEST WORD
	BR	TRUE	: CONTINUE
MOPUP	ADD	#24,SP	: REINSTATE SP
	RTS	R5	: RETURN
TSTBIT	000001		: TEST OK BIT

All integers above are octal.

APPENDIX F

Rough Comparison of In-Line Code with MICRO-MACRO Scheme

A typical test on a datum would be to check if it falls between an upper and a lower limit. Code for this would be similar to the following:

MOV DATUM, AREA	: MOVE DATUM TO SCRATCH AREA
CMP AREA, LOWLIM	: COMPARE TO LOWER LIMIT.
BGT UL	: IF GT GO TO UPPER LIMIT TEST
JMP FAIL	: IF EQUAL TO OR LT GO TO FAIL
UL CMP AREA, UPLIM	: COMPARE TO UPPER LIMIT
BLT SAVE	: IF LT SAVE RESULT
JMP FAIL	: IF GT OR EQUAL GO TO FAIL
SAVE ○	: TAKE STEPS TO STORE RESULTS FOR
○	: FUTURE USE.
○	
○	
JMP NEXTST	: GO TO NEXT TEST
FAIL ○	: FAIL ROUTINE
○	
○	

In addition to the above code are the entry and exit overhead costs. The structure of the MICROTTEST program is very similar to that shown here. The time required to execute the above code assuming that the datum falls within the window is about 10 μ s. Time required to save results depends on the program design; an average figure might be another 10 μ s, or 20 μ s total. Estimated time for the MICRO-MACRO scheme is 30 μ s. It is important to remember that the efficiencies of the MICRO-MACRO scheme become most apparent when used for intricate testing or where many tests are involved. Then the total times required are nearly equal or are larger for the In-Line scheme. The flexibility and generality of MICRO-MACRO are almost completely lacking in the In-Line method.

APPENDIX G

HISTOGRAMMING SCHEME

Histogram definitions are stored in tabular form for each level A, B, or C. Table contents for each histogram are as follows:

MACROTEST	
NAME	M
W1	W2
...	
W _{M-1}	W _M
LOWER LIM	W1
UPPER LIM	W1
MASK	W1
SCALE	W1
...	
LOWER LIM	W _M
UPPER LIM	W _M
MASK	W _M
SCALE	W _M

The word labelled Macrotest contains one or more bits set that correspond to the MACROS an event must pass in order to be histogrammed.

MACROTEST = Macrotests to be passed
 NAME = Name of Histogram
 W1, etc. = Relative address of words on which Histogram is based

SCHEMATIC HISTOGRAM ROUTINE