GULF GENERAL ATOMIC

Gulf-GA-B12354

A COMPUTER PROGRAM PACKAGE FOR THE FORT ST. VRAIN PULSED NEUTRON MEASUREMENTS

by

F. W. Todt

Prepared under
Contract AT(04-3)-633
for the
San Francisco Operations Office
U.S. Atomic Energy Commission

This document, which was prepared primarily for internal use, may contain preliminary or incomplete data. It is informal and is subject to revision or correction; it does not, therefore, represent a final report.

Gulf General Atomic Project 900 Date Published: January 2, 1973

GULF GENERAL ATOMIC COMPANY
P.O. BOX 81608, SAN DIEGO, CALIFORNIA 92138

ABSTRACT

Pulsed neutron experiments will be performed on the Fort St. Vrain reactor prior to startup to determine the subcritical reactivity of the core with various control rod configurations. The Fort St. Vrain Data Logger computer will be used to record and process the data. This report documents the computer program package which will be used in the conduct of these experiments.

CONTENTS

FIGURES

TABLES

1. INTRODUCTION

Pulsed neutron experiments will be performed on the Fort St. Vrain reactor prior to startup to determine the subcritical reactivity of the core with various control rod configurations. The reactivity obtained from these experiments will be compared to previously calculated values as a check on the nuclear design and will provide data for the subsequent operation of the reactor.

In the pulsed neutron experiments, bursts of fast neutrons are repetitively introduced into the subcritical assembly and thermal flux is monitored as a function of time. The Fort St. Vrain Data Logger computer will be used to record and process the data from each of three neutron detectors. By including a computer in the procedure the requirement for multichannel analyzer equipment is eliminated and the experimental reactivities and other information will be available to the experimenters immediately after each experiment without further effort. The raw data will also be recorded by the computer on paper tape for more sophisticated analyses at a later time.

2. OVERVIEW OF THE PULSED NEUTRON MEASUREMENTS COMPUTER PROGRAM PACKAGE

2.1. PURPOSE

The pulsed neutron experiment program package provides for the collection and storage of raw count data from three neutron detectors connected as inputs to a computer, and for the determination of the prompt neutron decay constant by the computer upon operator demand. No multichannel analyzer is required for the experiment. The raw data can be recorded on punched paper tape for further analysis later. A print out of all raw count data is also available.

2.2. EQUIPMENT USAGE

The experimenter controls the use of the program package through the operators console located in the control room.

The equipment to be used for the pulsed source experiments consists of a pulsed neutron source, an external timing device, and 3 neutron detectors with associated electronics which are connected to 5 interrupt lines of a Control Data 1700 computer (the Fort St. Vrain Data Logger). This equipment and the associated computer peripheral devices of interest to the experimenter are shown in Fig. 1.

The external timer is required for generating interrupt signals to the computer to divide the time period between neutron pulses into equal length time channels. Both the external timer and the neutron pulse generator frequencies may be set by the experimenter. Counts recorded by the detectors also generate interrupt signals to the computer and interrupt processing software then causes the counts and pulses to be recorded by the computer and ultimately stored on a mass storage device. Programs as well as data

Publ 2354/1

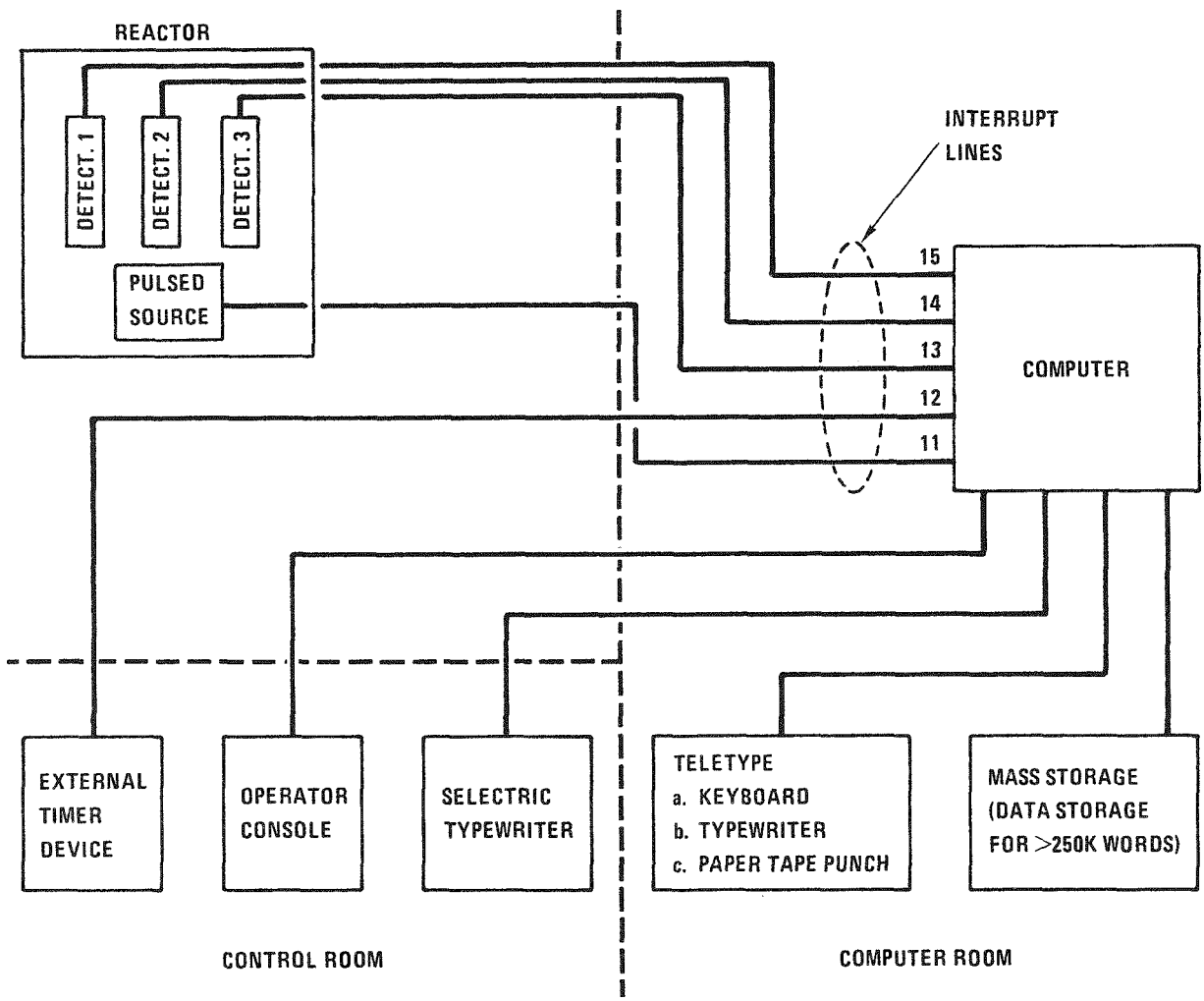

Fig. 1. Interconnection of equipment to be used for the pulsed neutron experiments

are stored on mass storage. The programs are brought into computer memory in response to signals from the operator console. Once the experiment has been set up, the experimenter need only manipulate buttons and switches at the operator console to take data and to process the data obtained.

Information display is by printed messages on a typewriter in the control room. After completion of an experiment, the raw data may be punched on paper tape, using the paper tape punch on the teletype located in the computer room.

2.3. PROGRAM USAGE

The program package for the pulsed source experiments consists of eight tasks or functions, packaged into several physical programs:

Task 1 - Begin data collection	- Initialize storage and begin recording count data from pulsed source experiment
Task 2 - Continue data collection	- Add to data previously recorded
Task 3 - Print raw count data	- Print raw count data by channel for each detector
Task 4 - Calculate α - ranges preset	- Perform α calculation using prompt and background channel range data from mass storage
Task 5 - Calculate α - determine ranges	- Perform α calculation using channel ranges based on the count data for a specified detector
Task 6 - Set "accept data" mode	- Data will be accepted for storage by Task 8

Task 7 - Set "print" mode - Data in storage will be printed by Task 8

Task 8 - Manual input function - Data will be accepted or printed for designated item depending on last mode set by Task 6 or 7

Selection of tasks is accomplished by the demand function switch position. Tasks may be selected in any sequence and as often as required. In the usual operation sequence, certain parameters would be entered using Task 8. Then counts would be recorded for a period of time by Task 1 or 2. Task 3 allows a flexible amount of print out to be taken and this would then be run followed (perhaps) by Task 4 or 5 to obtain the prompt neutron decay constant and reactivity. The experimenter can judge when and how often any program should run from the information printed by the computer.

3. OPERATING INSTRUCTIONS FOR PULSED NEUTRON MEASUREMENTS PROGRAM PACKAGE

3.1. OPERATOR CONSOLE

1. Demand Function Selector Switches - the setting of this pair of rotary switches determines the program to be run.

2. "ENTER" Button - used to cause computer to begin running a program. (Tasks 1, 2 and 3 may also be terminated by depressing this button.)

3. Thumbwheels - used to enter data into the computer. There are two sets of eight rotary switches aligned horizontally. The left most switch of each set permits the entry of a + or - sign. The right most switch of each set is marked for use as a multiplier. Neither of the sign nor multiplier switches are used by the tasks of the pulsed neutron measurements package. The other six switches allow the entry of numeric data values up to 32767.

3.2. COMPUTER ROOM

The computer room contains a teletypewriter, Control Data 1700 computer and a mass storage unit. In order to punch data tapes it will be necessary to have access to this equipment. The procedure for punching tapes is detailed later.

3.3. OPERATING PROCEDURE

To begin an experiment, three neutron detectors, an external timer, and a neutron pulse generator must be in position, be running, and have their outputs connected to the computer. These inputs to the computer are required only for the duration of the experiments. Once the equipment is operating, the tasks to be executed are selected by following the procedure in the next section. The results appear on a typewriter located in the

control room. After each experiment, a paper tape containing the data may be punched for further analyses.

3.4. PULSED NEUTRON MEASUREMENTS TASK ACTIVATION

To run a program, select the demand function number of the task by setting the rotating switches, manipulate the thumbwheel settings to enter the desired parameters, then press the "ENTER" button. Tasks 1 and 2 (MCA) do not terminate automatically (the others do). To stop MCA, depress "ENTER" again. Task 3 (PRINT) may also be terminated this way. However if it runs to completion then depressing "ENTER" will cause another task to be run.

Task	Activity	Demand Function No.	Program Name	Thumbwheel Setting	
				Left	Right
1	Begin data collection	51	MCA	0	--
2	Continue data collection	51	MCA	1	
3	Print raw count data	52	PRINT	Channel step	Last channel to print
4	Calculate α (ranges preset)	56	ALCALC	0	--
5	Calculate α (determine ranges from detector N)	56	ALCALC	N	--
6	Set "accept data" mode	53	MANIN	1	0
7	Set "print" mode	53	MANIN	1	1
8	Manual input function (see Table 1)	53	MANIN	Data Item No.	Data value (no effect in print mode)

3.5. TYPEWRITTEN MESSAGES

Task 1 - Begin data collection

BEGIN MCA

MCA OFF

TABLE 1
MANUAL INPUT DATA ITEMS FOR PULSED NEUTRON EXPERIMENTS

Item No.	Data
1	Manual input program mode: 0 - accept data 1 - print data
2	First channel to be printed (Task 3)
3	Not used
4	First channel in prompt decay range
5	Last channel in prompt decay range
6	First background channel
7	Last background channel
8	Number of groups of prompt channels for alpha calculation (≤ 8)
9	Dead time for detector (nanoseconds)
10	Channel width (microseconds)
11	First smoothing criterion x 10^3 (Try 690)
12	Second smoothing criterion x 10^3 (Try 0)
13	0 - Channel ranges preset (input items No. 4, 5, 6, & 7 1 - Determine channel ranges from data of detector 1 2 - Determine channel ranges from data of detector 2 3 - Determine channel ranges from data of detector 3
14, 15, 16, 17, 18	Not to be used
19	0 - skip data smoothing, 1 do data smoothing
20	Minimum count correction factor x 10^3 (Try 10)
21	Delayed neutron fraction x 10^6
22	Neutron generation time (microseconds)

Task 2 - Continue data collection

MORE MCA

MCA OFF

Task 3 - Print raw count data

CHANNEL	COUNT.1	COUNT.2	COUNT.3
1	0	0	0
.			
.			
.			

etc. to channel 400

NO. OF PULSES = 0

Task 4 - Calculate α - ranges preset

PROMPT CHANNELS		GRP WIDTH	B.G. CHANNELS		PULSES	DETECTOR
19	120	17	140	200	9999	0

DETECTOR	1		2		3	
B.G.COUNTS	2778		0		0	
GRP	ALPHA	ERROR	ALPHA	ERROR	ALPHA	ERROR
1	-.316+02	.29-01	.000	.00	.000	.00
2	-.324+02	.14-01	.000	.00	.000	.00
3	-.329+02	.21-01	.000	.00	.000	.00
4	-.336+02	.40-01	.000	.00	.000	.00
5	-.355+02	.77-01	.000	.00	.000	.00
6	-.389+02	.16+00	.000	.00	.000	.00
AVG	-.326+02	.10-01	.000	.00	.000	.00
RHO	-.764-02		.000		.000	
S	-.121+01		.000		.000	
K	.992+00		.000		.000	

LAMDA	BETA	DELTA	TAU	CORREX	SMOOTH1	SMOOTH2
490	6500	1000	2000	10	690	0
-6	-6	-6	-9	-3	-3	-3

Task 5 - Calculate α - determined ranges

Similar to Task 4 except FLAG $\neq$ 0

Task 6 - Set accept data mode

ACCEPT DATA MODE

Task 7 - Set print mode

EDIT ONLY MODE

Task 8 - Manual input function

```
                ITEM  3      VALUE =    7
                ITEM  4      VALUE =    8
                ITEM  5      VALUE =   82
                ITEM  6      VALUE =  113
                ITEM  7      VALUE =  321
                ITEM  8      VALUE =    5
                ITEM  9      VALUE = 2000
                ITEM 10      VALUE =  330
                ITEM 11      VALUE =   10
                ITEM 12      VALUE =    3

ACCEPTED ITEM 17      VALUE =   99
ACCEPTED ITEM 18      VALUE =   99
ACCEPTED ITEM 19      VALUE =   99
```

Diagnostic Comments

TRY AGAIN

CHECK INPUT - TRY AGAIN

TOO MANY BAD CHANNELS

TOO MUCH BACKGROUND

CHECK THUMBWHEEL SETTING

BAD DATA

**** PUNCH PAPER TAPE IF FINISHED

CHECK THUMB WHEELS --- TRY AGAIN

3.5.1. Interpretation of the Typewritten Messages

Tasks 1 and 2

The first message appears when count data is being recorded and the second message appears when data collection has been terminated.

Task 3

CHANNEL refers to the time intervals. COUNT.N refers to the Nth detector and the data printed beneath each is the number of counts accumulated by that detector.

Tasks 4 and 5

The table below indicates the meaning of the labels used in the alpha calculation printout.

Label	Meaning
PROMPT CHANNELS	First and last time channel considered in prompt neutron range
GRP WIDTH	Number of channels in a small group of channels within the prompt neutron range for which α is calculated ($10 \leq$ GRP WIDTH ≤ 25)
B.G. CHANNELS	First and last time channels considered to determine background count level
PULSES	Total number of neutron pulses recorded during experiment
FLAG	Value of 0 indicates prompt and background channel ranges used in calculation were obtained from stored data Value of 1, 2 or 3 indicates data recorded by one of the detectors was used to determine prompt and background ranges
DETECTOR	Each neutron detector is designated by a number
B.G. COUNTS	Average background count level over background channel range

Label	Meaning
GRP	Number designating a small group of time channels in the prompt neutron decay time range
ALPHA	Prompt neutron decay constant
ERROR	A measure of the uncertainty in ALPHA
RHO	Subcritical reactivity
$	Reactivity as a multiple of delayed neutron fraction
K	Effective multiplication factor
LAMDA	Neutron generation time (seconds)
BETA	Delayed neutron fraction
DELTA	Width of a time channel in seconds
TAU	Detector dead time (seconds)
CORREX	Minimum count correction factor
SMOOTH1	First data smoothing criterion
SMOOTH2	Second data smoothing criterion

Task 6

Message indicates that the use of task 8 will cause data to be stored on mass storage.

Task 7

Message indicates that use of task 8 will produce a printout of a value on mass storage.

Task 8

The first form of the message indicates the value in mass memory of the table item indicated. The second form of the message is preceded by the word "ACCEPTED" and indicates that a value has been read from the thumbwheel switches and has been stored into the table in mass memory. Normally the two forms of message cannot appear together because the mode must be changed in between using task 6 or 7.

Diagnostic Comments

However "TOO MUCH BACKGROUND" indicates that all channels appear to be in the background range. "BAD DATA" indicates that the prompt channel range appears to contain less than 10 channels.

3.6. NOTES ON USE OF THE PROGRAMS

3.6.1. MCA (Tasks 1 and 2)

This program stores zeros in all channels when run in "initialize" mode. Don't forget to set the left thumbwheels to 1 when in "continue" mode. Program execution can be stopped at any time by depressing the "ENTER" button. Do not press it twice in quick succession.

3.6.2. PRINT (Task 3)

This program will print counts for all channels from 1 to 400 after the experiment but this takes 17 minutes. During the experiment it may be best to print only data in the prompt decay range (say up to channel 100) in steps of 5 or 10 time channels to keep print time to about 30 seconds. The first channel to be printed should be preset by storing the channel number into data item 2 of the manual input table with the use of Task 8. Printing of the count data may be aborted by depressing the "ENTER" button while there is still more printing to be completed.

3.6.3. ALCALC (Tasks 4 and 5)

The program requires that the values of certain parameters be available on mass storage before any attempt is made to run the code (see Task 8 and Table 1). Prompt and background ranges (in terms of time channels) may be determined by the code using the data from one of the detectors, or these ranges may be preset. It is expected that for the first few executions, the ranges would be calculated, and thereafter arbitrarily fixed so that the data can be compared directly with previous runs. The program requires about 2 minutes of execution time.

3.6.4. MANIN (Tasks 6, 7, and 8)

Task 8 operates in either the "print" or "accept data" mode. The mode is set by running Tasks 6 or 7 (that is by storing a zero or a one in item 1 of the data list). Either mode specified remains effective until changed again. Data is entered thru the use of the thumbwheels on the operator console and is scaled internally where necessary. The manual input data table for all pulsed source experiment programs is given in Table 1.

3.7. PUNCHING PAPER TAPE

A program to initiate paper tape punching from the operator console will probably be available. If not, then to punch a paper tape containing data from a pulsed source experiment, the following procedure may be followed:

1. Bring the "Olympus" utility program into the computer

    a. Mass storage controller - place "TEST" switch UP

    b. Computer console - move "CLEAR" switch (momentarily) and return to center position

    c. Mass storage controller - depress "AUTOLOAD" button

    d. Computer console - move "RUN/STEP" switch (momentarily) to "RUN" and return to center position

    Bell rings on teletype and Olympus is in computer.

2. Get about one foot of leader on paper tape

    a. Teletype unit - press "T" button (3/4" diam.)

b. Teletype unit - key simultaneously CNTR
SHIFT
P
REPT

until enough leader is present.

Label the tape leader using white or yellow pencil.

3. Read data from mass storage into core.

a. Mass storage controller - place "TEST" switch UP

b. Teletype unit - type: RDM, 100, 5B0, 7, 1CR (CR is carriage return)

4. Enter punch tape instructions

a. Teletype unit - press "K" button

b. Teletype unit - type: PPT, 100, 5B0 CR (CR is a carriage return)

c. Teletype unit - press "TTR" button

d. Computer console - move "SKIP" switch down (momentarily) and return to center position.

Tape begins punching.

5. After punching is finished

a. Teletype unit - press "K" button

b. Computer console - move "SKIP" switch momentarily and return to center position.

Do <u>not</u> attempt to bring the operating system in until tape punching has been completed.

6. Bring operating system "MSOS" in

   a. Mass storage controller - place "TEST" switch DOWN

   b. Computer console - move "CLEAR" momentarily and return to center position.

   c. Mass storage controller - depress "AUTOLOAD" button

   d. Computer console - move "RUN/STEP" switch (momentarily) to RUN and return to center position

   e. Teletype unit - key in 6 digit date and CR (day, month, year- no spaces or commas)

   f. Teletype unit - key in 4 digit time and CR (hours, minutes- no spaces or commas)

A typewriter in the control room will print a message verifying that MSOS is in. Now programs can be run for the next experiment.

4. THE PROGRAMS

4.1. PROGRAM MCA

4.1.1. Function

The program records the counts from three neutron detectors and stores the data on mass storage. Count data collection can be started, stopped, and continued as the experimenter desires.

4.1.2. Method

The procedure followed in execution of the program is relatively uncomplicated and is shown in Fig. 2, Flow Diagram of the MCA Program.

During the pulsed neutron experiments, the interrupt system of the computer is temporarily connected to the counter electronics of three neutron detectors, an external timer, and a neutron pulse generator. All other interrupts to the computer (except one from the operator console) are inhibited by the program because any other computer activity would interfere with the time critical nature of the data being taken. A signal from the neutron pulse generator interrupts the computer at the beginning of each pulse. A second signal generated by the external timer divides the time between neutron pulses into a number of equal time intervals (channels) into which the counts from the detectors are recorded. The program allows for a maximum of 400 time channels. Therefore, the ratio of the frequency of the external timer to the frequency of the neutron pulse generator should be approximately 400. A higher frequency ratio will generate a larger number of time channels and all counts in the additional channels would be lost. A lower ratio will decrease the number and, of course, widen the channels unnecessarily.

Although the computer can recognize relatively short interrupt signal pulses (approximately 500 nanoseconds), the width of the interrupt signals are required to be considerably longer. For this method of collecting data to work, the following criteria must be satisfied:

1. All interrupt signals should be as long as possible to maximize the possibility of recognition in competition with the other signals.

2. All interrupt signals (hardware) should be shorter than their own interrupt processor (software) execution times. Violation of this criterion would result in processing the interrupt more than once for each signal.

3. Higher priority interrupt signals should be longer than lower priority interrupt processors so that a high priority interrupt will never be missed because the computer was processing a lower priority interrupt at the time.

4. The required priority of interrupts from highest to lowest is: pulsed source, external timer, neutron detectors.

Table 2 gives the extended interrupt signal widths required to meet the above criteria.

TABLE 2
SPECIAL INTERRUPT SIGNALS

Interrupt Line	Signal Source	Interrupt Processor Execution Time (μ sec)	Interrupt Signal Width (μ sec)
11	Pulsed source	14.3	13.5 ± 0.5
12	External timer	12.1	11.5 ± 0.5
13	Detector 1	8.8	8.0 ± 0.5
14	Detector 2	8.8	8.0 ± 0.5
15	Detector 3	8.8	8.0 ± 0.5

The experiment data is stored in the computer and on mass storage as integer values in order to economize on storage and execution time. The 16 bit word length of the Control Data 1700 however limits the magnitude of the data values to 32767. For any given pulse frequency then, the maximum counting time can be determined in order not to overflow the value for the number of pulses. It will also be necessary to check the number of counts recorded per channel periodically so they do not overflow. The rate of count accumulation can be checked using the print function (Task 3).

The MCA program normally spends most of its execution time looping thru a series of instructions which do nothing at all (the "idle" loop). When an interrupt signal appears, the computer automatically causes a jump to a series of locations called an "interrupt trap" and processes the instructions provided for the particular interrupt involved. Interrupts from the pulsed source equipment cause the channel index to be reset to 1. Interrupts from the external timer cause the channel index to be incremented. An interrupt from a detector causes the count to be recorded (added) in the channel determined by the value of the channel index. The computer then returns to the "idle" loop and effectively does nothing more until it receives another interrupt.

To terminate the count collection procedure, the experimenter need only to press the "ENTER" button on the operators console. This generates an interrupt on line 7 and the interrupt processor sets a flag which the program recognizes as a signal to remove the program from the computer.

4.2. PROGRAM PRINT

4.2.1. Function

The program provides the capability to print out the count data which has been recorded for each of the three detectors. By entry of data through

the thumbwheels on the operators console the experimenter may choose to print every Nth channel from the first channel designated to the last channel designated. A procedure to abort the printing by depressing the "ENTER" switch has been included.

4.2.2. Method

The execution sequence is indicated in Fig. 3, Flow Diagram of Program PRINT. The program consists of the statements required to read the thumbwheel data, pick up the count data from drum, and print in the manner requested.

4.3. PROGRAM ALCALC

4.3.1. Function

The program processes the raw data taken during a pulsed neutron experiment and prints the prompt neutron decay constant and subcritical reactivity on a typewriter located in the control room. It does not change the raw count data stored in the computer system. Therefore data collection can be continued after this program has been run.

4.3.2. Method

Raw count data in integer form which has been collected and stored by the MCA program is read into the computer and converted to floating point form. After correcting for counting losses and computing weights for the counts in every channel for each detector, data smoothing may optionally be performed. The objective is to eliminate data produced by noise signals which are obviously not valid data. The background amplitude, assumed to be constant for all channels, is then obtained by averaging counts from the last channel with counts recorded and working forward toward the prompt decay range. Next, the prompt range is determined. The lower end of this range is taken to be at the channel where the count is 3 times the background amplitude.

Alpha, the prompt neutron decay constant, and alpha uncertainty is then calculated from the results of a linear least squares fit of the log of the corrected count data less the background. Alpha is computed for an arbitrary number (1-8) of groups of channels in the prompt range and for the entire prompt range. Reactivity is then computed using alpha from the entire prompt neutron decay range. The calculation takes about 2 minutes. The results are then printed.

Determination of the prompt and the background channel ranges is controlled by options selected at the operators console. The program may use data stored on mass storage (by previous calculation or by manual insertion by the operator) for the channel ranges, or the ranges can be determined by the program based on the count data collected from a designated detector.

Figure 4 shows the calculation sequence of the ALCALC program and Fig. 5 is a flow diagram of the portion of the program which prints the results.

4.3.3. Details of the Calculation Procedure

1. Read input parameters from mass storage and thumbwheels.

   Some data are always required to be input through the operator's console before attempting to use the code. These are data items 8, 9, and 10 of the manual input data table: number of channel groups, dead time for the detectors, and the channel width respectively. The left hand thumbwheels at time of activation of the program contain the channel boundary option selection. Items 4, 5, 6, and 7 of the manual input data table, the prompt and background channel boundaries, may be determined by the code during the run, data generated in the previous run may be used, or the data may be entered by hand.

2. Compute nominal time position for each channel

$$t_i = \left(i - \frac{1}{2}\right) \Delta$$

where i = channel number

Δ = channel width (μsec)

3. Correct raw count for dead time

$$C_i = \frac{C_i^{RAW}}{1 - \frac{C_i^{RAW} \cdot \tau}{P\left(\Delta + \frac{\tau}{2}\right)}}$$

where τ = dead time during which a count cannot be recorded (input)

C_i = corrected counts

C_i^{RAW} = raw count data

P = total number of neutron pulses

Skip count correction if:

$$C^{RAW} < \frac{\varepsilon_x}{\frac{\tau}{P\left(\Delta + \frac{\tau}{2}\right)}}$$

where ε_x = minimum count correction factor (fraction of raw counts)

4. Compute weights for corrected counts

$$\sigma_i = \frac{\sqrt{C_i^{RAW}}}{C_i}$$

$$w_i = \left(\frac{1}{\sigma_i}\right)^2$$

where w_i = weight assigned when performing least square fit

5. Take natural log of counts in each channel and smooth the data. Set weight of bad channels to zero and the count data to the fit value.

Data smoothing is optional and may not be needed. This procedure fits the log of the corrected count data to a straight line (in groups of 25 channels). The data is then tested to determine if:

$$|\ell n\ C - \ell n\ C^{Fit}| < \varepsilon_s$$

where ε_s is a smoothing criterion

Data meeting the test is assumed to be valid and data failing the test is arbitrarily set to the fitted value and weighted by zero.

Two smoothing passes are made if two values are given for the smoothing criterion. Very large deviations which affect the fit can be eliminated on the first pass and then a tighter criterion can be used on the second pass to complete the smoothing.

The calculation is aborted if more then 30% of the data is found to be bad and a comment to that effect is printed.

6. Determine background range using data from specified detector. Then average counts for 25 channels.

$$BG_N = \frac{\sum_{i=M2}^{M1} C_i}{25}$$

where BG_N = average background count for Nth group of 25 channels
M2 = last channel in group of 25 channels
M1 = first channel in a group of 25 channels

Working in groups of 25 channels, towards the prompt range from the last channel with data, obtain a running average of the background count over N channel groups.

$$BGBASE_N = \frac{BG_N + (N-1)BGBASE_{N-1}}{N}$$

Terminate the procedure when $|BG_N - BGBASE_{N-1}| > \sqrt{2\ BGBASE_{N-1}}$ and assume that the first channel of channel group N-1 is the first channel of the background range.

7. Average the background count for each of the three detectors over the range of background channels. If channel is determined to have bad data, then the value assumed for background averaging is the fit value.

8. Determine prompt neutron decay range from data of the specified detector. Beginning with channel 1, search for maximum count. This is the first channel for prompt range.

   To locate the last channel of the prompt range the count data is fit to straight lines in groups of 20 channels. When the count predicted by the fit is less than 3 times the background count that channel is used as the last channel in the prompt range.

   The prompt range is then divided into the designated number of channel groups for purposes of the alpha calculation.

9. Subtract background from counts, take log and do least squares first order fit to obtain α and α uncertainty for each channel group in the prompt range.

$$\alpha_N = \begin{vmatrix} \Sigma w_i & \Sigma w_i C_i \\ \Sigma w_i t_i & \Sigma w_i t_i C_i \end{vmatrix} \Big/ D$$

$$\Delta\alpha_N = \sqrt{\frac{\sum_{i=1}^{J} W_i \left(\ell n\ C_i - \ell n\ C_i^{FIT} \right)}{J - 2} \left[\frac{\sum_{i=1}^{J} W_i}{D} \right]}$$

where J = number of channels in a channel group

C_i^{FIT} = count value determined by fit

$$D = \begin{vmatrix} \Sigma w_i & \Sigma w_i\ t_i \\ \Sigma w_i\ t_i & \Sigma w_i\ t_i^2 \end{vmatrix}$$

10. Compute α and α uncertainty over entire prompt range.

$$\alpha = \frac{\sum_{N=1}^{K} W_N\ \alpha_N}{\sum_{N=1}^{J} W_N}$$ total prompt neutron decay constant

$$W_N = \left(\frac{1}{\Delta\alpha_N} \right)^2$$ weight of a channel group

$$\Delta\alpha = \sqrt{\frac{1}{\sum_{N=1}^{K} W_N}}$$ total prompt decay constant uncertainty

11. Compute ρ, \$, Keff

$$\rho = \beta + \Lambda\alpha$$

$$\$ = \frac{\rho}{\beta}$$

$$\text{Keff} = \frac{1}{1 - \rho}$$

where ρ = reactivity

$ = reactivity in dollar units

Keff = effective multiplication factor

α = prompt neutron decay constant over entire range

β = delayed neutron fraction

Λ = neutron generation time

12. Print results:

α, $\Delta\alpha$, ρ, $, Keff, background for each detector, prompt and background channel ranges, applicable input constants.

4.4. PROGRAM MANIN

4.4.1. Function

This program provides for the entry of data into a special table by the use of the thumbwheels on the operator console. The data is stored in a reserved area of mass storage. Other programs may read or write data from or into this table. The user then has some freedom in controlling the operation of the programs since he can modify a large group of parameters in this way.

4.4.2. Method

The first item in the table of data is a special item which controls the execution mode. Data is read from storage and printed, one item at a time, or new data is stored depending on the contents of item 1. Errors in manipulating the thumbwheels are indicated when possible.

The execution sequence is shown in Fig. 6, Flow Diagram of Program MANIN. A table of the data items applicable to the pulsed source experiments program package is given in the operating instructions.

APPENDIX A

FLOW DIAGRAMS

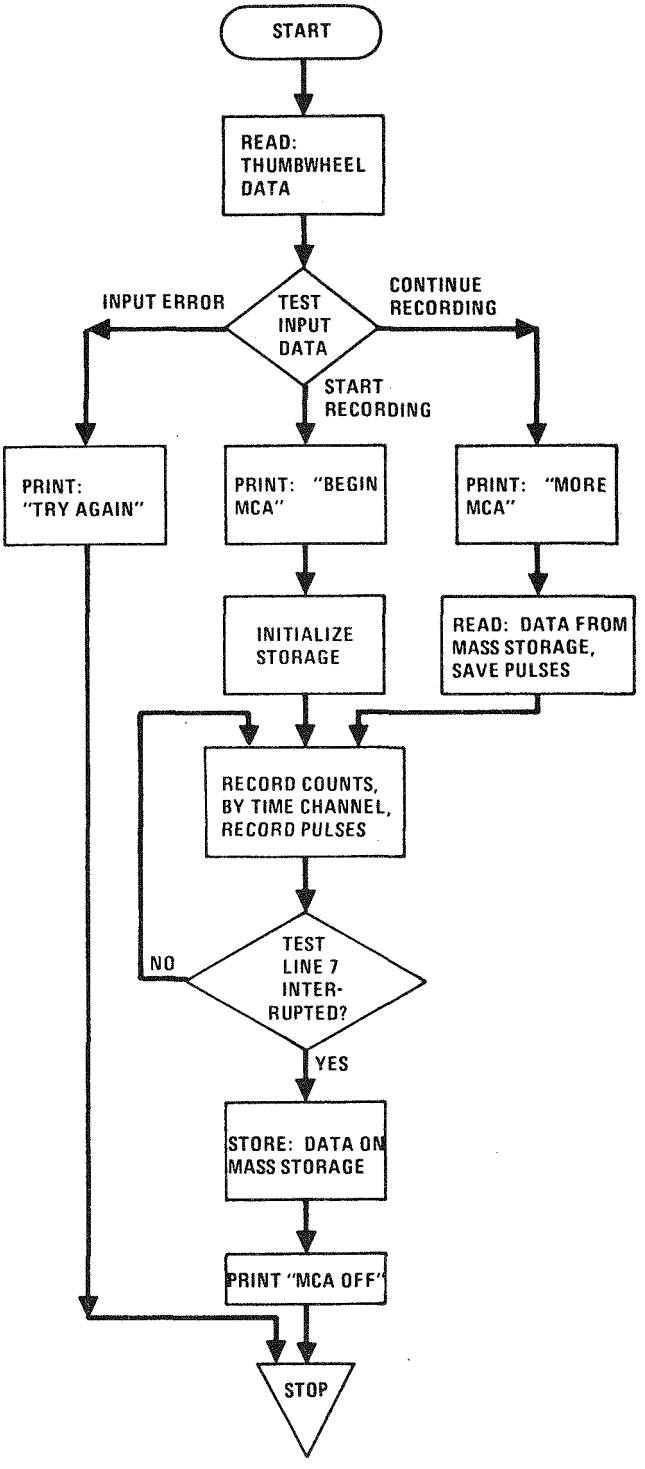

Fig. 2. Flow diagram of the MCA Program

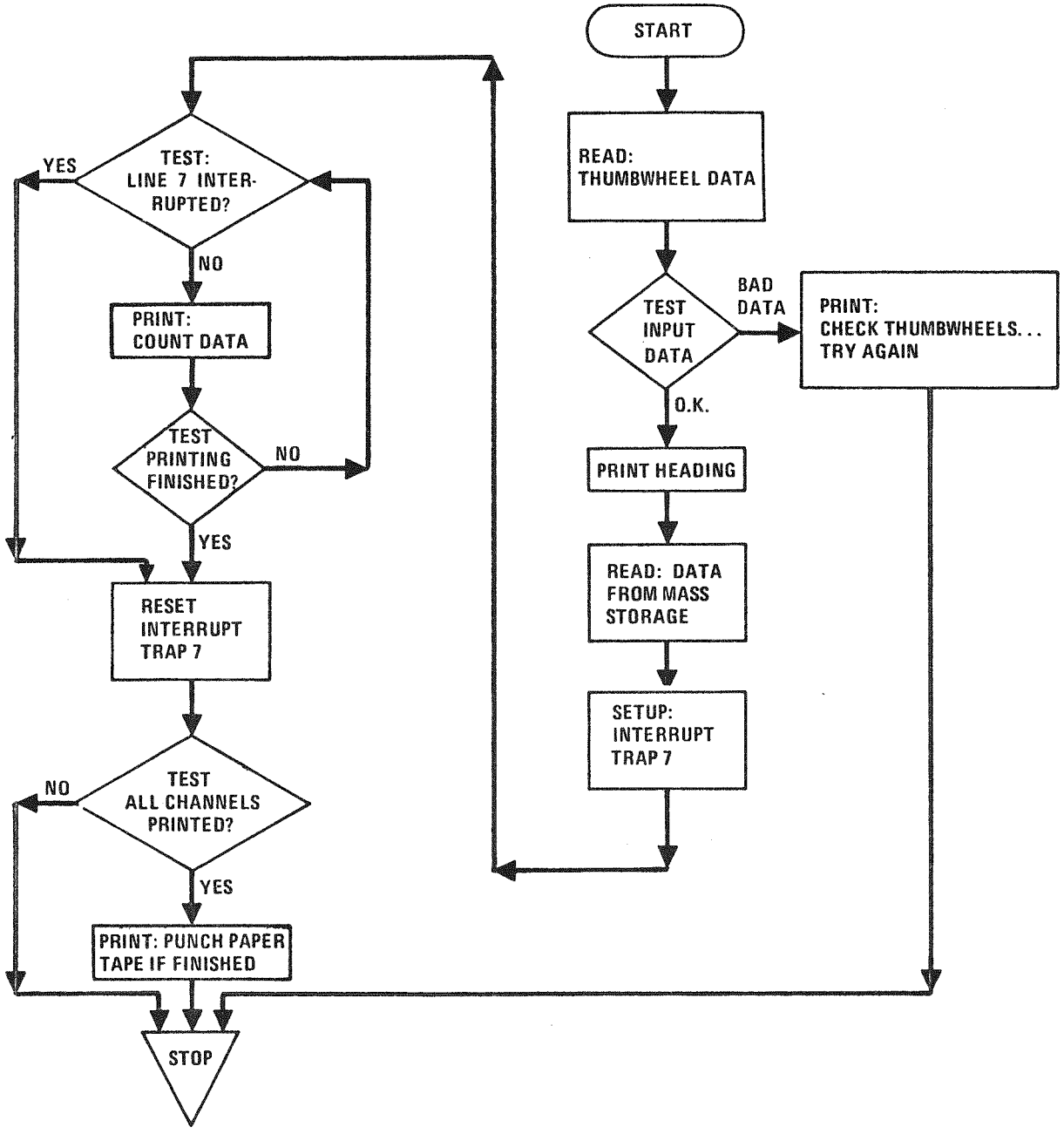

Fig. 3. Flow diagram of Program PRINT

Pub12354/6

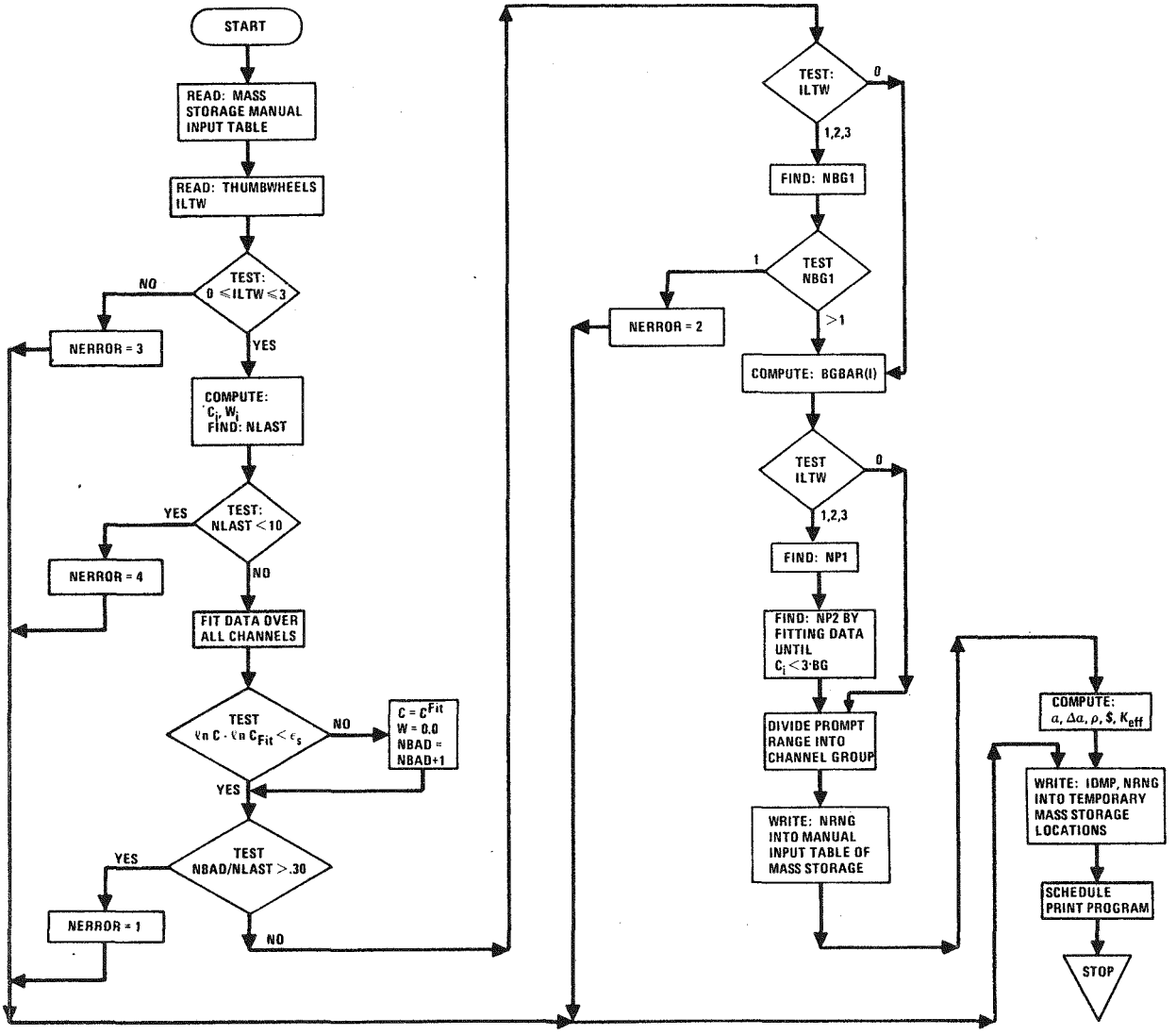

Fig. 4. Flow diagram of ALCALC Program

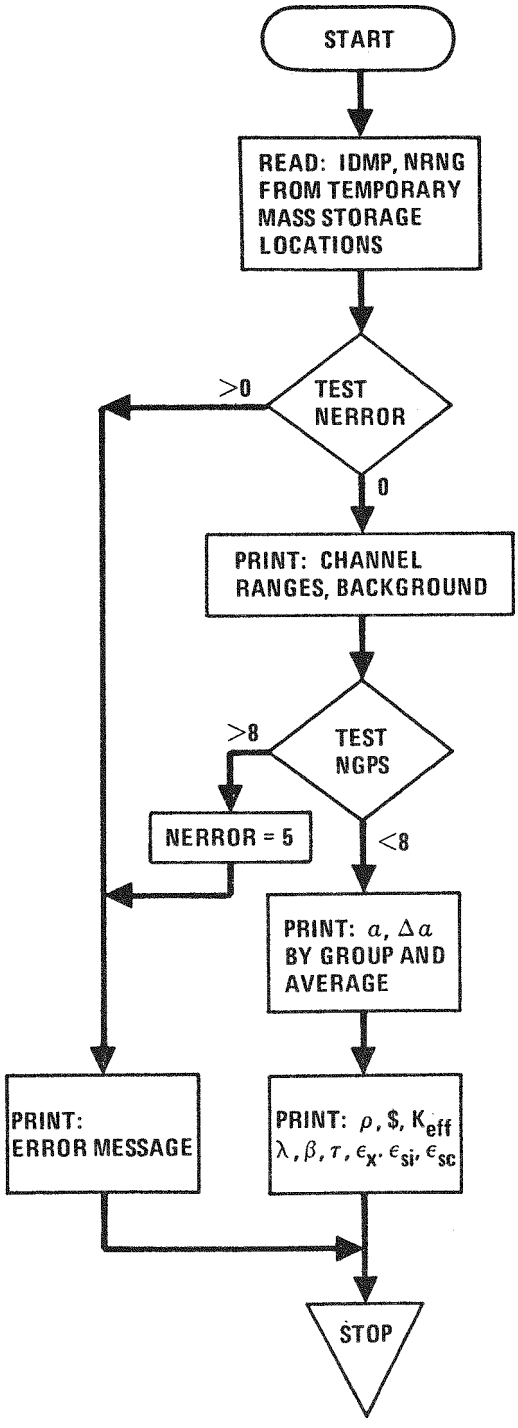

Fig. 5. Flow diagram of ALCALC Print Program

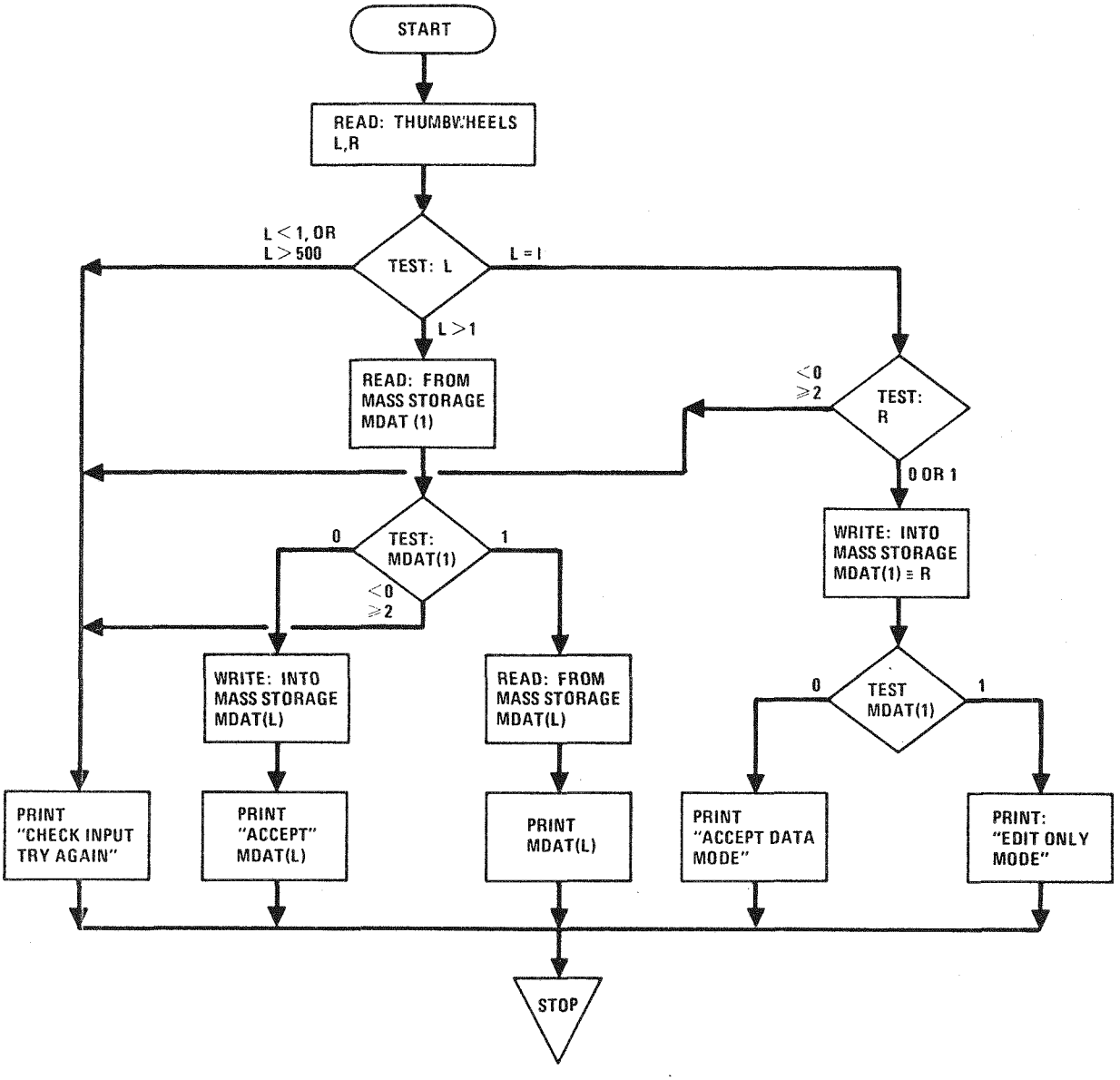

Fig. 6. Flow diagram of Program MANIN

APPENDIX B
SUBROUTINE LISTS

B.1. MCA SUBROUTINE LIST

1. BEGIN - Main program which controls the execution sequence primarily by making calls to other subroutines.

2. STATUS - Prints comments concerning program status.

3. DRUMIO - Sets up sequence of operations to read or write count data from or to mass storage.

4. MCAP - Loads interrupt traps, contains idle loop and special interrupt processors.

5. TRAPX - Contains entry points for unneeded library subroutines.

6. CODUM - Reserves block of common storage and eliminates hand patching of addresses.

B.2. PRINT SUBROUTINE LIST

1. FINAL - Main program which controls execution sequence primarily by making calls to other subroutines.

2. COMENT - Prints heading or comments.

3. TYPOUT - Prints count data.

4. DRUMIO - Reads data from drum.

5. TRAPX - Contains entry points for unneeded library subroutines.

6. SET7 - Sets up line 7 interrupt trap to cause jump to TRAP 7.

7. RSET7 - Restores line 7 interrupt trap.

8. TRAP7 - Interrupt processor for line 7; stores indication that ENTER button was depressed.

9. CODUM - Reserves block of common storage and eliminates hand patching of addresses.

B.3. ALCALC SUBROUTINE LIST

1. ALCALC - Main program controls the execution sequence, smooths the data, performs most of the calculations, and schedules ALPRN.

2. CWDATA - Writes or reads counts or weights on/from drum for a group of channels.

3. DRMDAT - Writes or reads blocks of data on/from mass storage.

4. FIT2 - Performs 1st order least squares fit.

5. TCHAN - Computes nominal time position of each channel.

6. TEST - Used for tracing program execution during debugging.

7. TRAPY - Contains entry points for unneeded library subroutines.

8. CODUM - Reserves block of common storage and eliminates hand patching of addresses.

B.4. ALRRN SUBROUTINE LIST

1. ALPRN - Main program to control execution of print sequence.

2. DRMDAT - Reads blocks of data from mass storage.

3. TRAPZ - Contains entry points for unneeded subroutines.

B.5. MANIN SUBROUTINE LIST

1. MANIN - Main program which controls the execution sequence primarily by making calls to other subroutines.

2. REMARK - Prints mode or error comment.

3. DRUMA - Reads/writes data from/to table on mass storage.

4. TRAPX - Contains entry points for unneeded library subroutines.

5. CODUM - Reserves block of common storage and eliminates hand patching of addresses.

APPENDIX C

SOURCE PROGRAM LISTINGS

C.1. MCA SOURCE PROGRAM LISTING

```
      PROGRAM BEGIN
C
C     MULTICHANNEL ANALIZER PROGRAM TO RECORD COUNTS
C
C     ICEL MUST LOCATED AT $3F00
      COMMON ICEL(255)
C
      DIMENSION IB(1200)
C
      RELATIVE  MCA,STATUS,DRUMIO,LINK,RELESE
C
C     SET MASK TO ACCEPT INTERRUPTS ON LINES 5,6 ONLY
C
      ASSEM     $0500,$C000,$0060,$0821,$0400
C
      NCH = 400
      IADR = LINK(0) + ICEL(1)
      ILTW = ICEL(IADR+13)
      IF (ILTW.LT.0) GO TO 1000
      IF (ILTW.GT.1) GO TO 1000
      IF (ILTW.EQ.0) GO TO 100
      IF (ILTW.EQ.1) GO TO 200
C
C     INITIALIZE DETECTOR DATA,RECORD COUNTS,WRITE DRUM
C
  100 CONTINUE
      CALL STATUS(1)
      DO 1 I=1,1200
    1 IB(I) = 0
C
      CALL MCA(IB)
      IB(400) = 0
      IB(800) = 0
      CALL DRUMIO(IB,1,NCH)
      CALL STATUS(3)
      GO TO 900
C
C     READ COUNT DATA FROM DRUM,RECORD COUNTS,WRITE DRUM
C
  200 CONTINUE
      CALL STATUS(2)
      CALL DRUMIO(IB,2,NCH)
      IACC = IB(1200)
      CALL MCA(IB)
      IB(400) = 0
      IB(800) = 0
      IB(1200) = IB(1200) + IACC
      CALL DRUMIO(IB,1,NCH)
      CALL STATUS(3)
C
  900 CONTINUE
C
C     RESET MASK TO ACCEPT INTERRUPTS ON LINES 5 THRU 10
```

```
C
      ASSEM     $0500,$C000,$07FF,$0821,$0400
C
      CALL RELESE(BEGIN)
C
 1000 CONTINUE
      CALL STATUS(*)
      GO TO 900
C
      END
```

```
      SUBROUTINE STATUS(NN)
C
      DIMENSION ITBL(3),ITEMP(8),IBUF(5)
C
      RELATIVE  ENCODE,FWRITE,DISPAT
      RELATIVE  Q8PKUP,Q8PREP
C
  100 FORMAT (10H1BEGIN MCA)
  200 FORMAT (10H1MORE MCA )
  300 FORMAT (10H1MCA OFF  )
  400 FORMAT (10H1TRY AGAIN)
C
      LU = $194F
      ASSIGN 500 TO ICOMP
      GO TO (11,12,13,14),NN
   11 ASSIGN 100 TO IFORM
      GO TO 20
   12 ASSIGN 200 TO IFORM
      GO TO 20
   13 ASSIGN 300 TO IFORM
      GO TO 20
   14 ASSIGN 400 TO IFORM
   20 CONTINUE
      CALL ENCODE(IBUF,IFORM,0,0)
      CALL FWRITE(LU,IBUF,5,ICOMP,$44,ITEMP)
      CALL DISPAT
  500 CONTINUE
C
C     LOOP UNTIL TYPEWRITER NOT BUSY
C
      ASSEM     $C400,$4B7,$A00C,$20,$111,$18FA
      ASSEM     $C400,$4B7,$A000,$20,$101,$18FA
      RETURN
      END
```

```
      SUBROUTINE DRUMIO(IB,NDO,NCH)
C
C     READ OR WRITE COUNT DATA ON DRUM  NDO=1-WRITE,2-READ
C
      DIMENSION IB(1200)
      DIMENSION ITBL(3),ITEMP(8)
C
      RELATIVE  WRITE,READ,DISPAT
      RELATIVE  Q8PKUP,Q8PREP
C
      LU = $941
      ITBL(1) = NCH
      ITBL(2) = 7
C
      ASSIGN 500 TO ICOMP
      DO 500 K=1,3
      L = (K-1)*NCH + 1
      ITBL(3) = L
      GO TO (1,2),NDO
    1 CALL WRITE(LU,IB(L),ITBL,ICOMP,$44,ITEMP)
      GO TO 3
    2 CALL READ (LU,IB(L),ITBL,ICOMP,$44,ITEMP)
    3 CALL DISPAT
  500 CONTINUE
      RETURN
      END
```

```
       NAM       MCAP
       ENT       MCA
       BZS       PULSE,TEST7,BACK,IBMAX,IBADR,TEMP
XX     ADC       QU,PS,TM,D1,D2,D3,MCA
*
MCA    NUM       0
       IIN
*
*      ESTABLISH ADDRESSES AT EXECUTION TIME
*
       LDA*      MCA
       STA*      TEMP                ABS LOC OF -DEL FROM ADC TO IB
       ADD*      (TEMP)              A REG NOW HAS ADR OF IB
       STA*      IBADR
*
       ENQ       5
FXLOC  LDA*      MCA
       SUB       =N1
       STA*      TEMP                ABS ADR OF DELTA
       ADD*      (TEMP)              A REG NOW HAS TRUE ADR OF MCA
       SUB*      XX+6                A REGISTER NOW HAS ADDR CORRECTION
       ADD*      XX,Q
       STA*      XX,Q
       INQ       -1
       SQM       1
       JMP*      FXLOC
*
*      SET UP INTERRUPT TRAPS
*
       LDA       =N$1C02             JMP*      (+2)
       STA+      $11D                FOR INTERRUPT 7     -   ENTER
       STA+      $12D                FOR INTERRUPT 11    -   PULSES
       STA+      $131                FOR INTERRUPT 12    -   CHANNEL
       LDA       =N$D600             RAO+      DX,Q
       STA+      $135                FOR INTERRUPT 13    -   DETECTOR 1
       STA+      $139                FOR INTERRUPT 14    -   DETECTOR 2
       STA+      $13D                FOR INTERRUPT 15    -   DETECTOR 3
*
       LDA*      XX                  LOC OF LINE  7  INTERRUPT PROCESSOR
       STA+      $11F
       LDA*      XX+1                LOC          11
       STA+      $12F
       LDA*      XX+2                LOC          12
       STA+      $133
       LDA       =N$0000             LOC FOR D1   IB(1)
       STA+      $136
       LDA       =N$0190             LOC FOR D2   IB(401)
       STA+      $13A
       LDA       =N$0320             LOC FOR D3   IB(801)
       STA+      $13E
       LDA       =N$0E34             EXI       $34
       STA+      $137
       LDA       =N$0E38             EXI       $38
```

```
       STA+      $13B
       LDA       =N$0E3C             EXI       $3C
       STA+      $13F
*
*      ESTABLISH LIMIT FOR DETECTOR DATA
*
       LDA*      IBADR
       ADD       =N400
       STA*      IBMAX
*
MASKA  LDA       =N$F880             MASK FOR INTERRUPTS 7,11,12,13,14,15
       TRA       M
       LDQ*      IBADR
       EIN
*
*      IDLE LOOP  - EXECUTES HERE EXCEPT WHEN PROCESSING INTERRUPTS
*
IDLE   NOP
       LDA*      TEST7
       SAN       1
       JMP*      IDLE                END OF IDLE LOOP
       JMP*      TERM                JMP TO PREPARE FOR TERMINATION
*
*      INTERRUPT PROCESSORS
*
QU     ENA       -1                  ENTER SWITCH
       STA*      TEST7
       LDQ       =N$400
       INP       -1
       LDQ       =N$9000
       INP       -1
       INQ       1
       INP       -1
       EXI       $1C
*
PS     RAO*      PULSE               PULSED SOURCE
       LDQ*      IBADR
       NOP
       NOP
       EXI       $2C
*
TM     INQ       1                   TIME SEG  (CHANNEL)
       TRQ       A
       SUB*      IBMAX
       SAM       1
       INQ       -1
       EXI       $30
*
D1     NUM       0                   DUMMY - D1 INTERRUPT REMOVED
*
D2     NUM       0                   DUMMY - D2 INTERRUPT REMOVED
*
D3     NUM       0                   DUMMY - D3 INTERRUPT REMOVED
*
```

```
*       TERMINATION PROCEDURE
*
TERM    IIN   0
TRAP    LDA         =N$54FE             RESET INTERRUPT TRAP7
        STA+        $11D
        LDA         =N$675
        STA+        $11F
MASKB   LDA         =N$60               MASK FOR LINES 5,6
        TRA         M
        EIN
*
PULL    LDA         PULSE               STORE TOTAL PULSES FOR THIS RUN
        LDQ         IBADR
        STA+        1199,Q              INTO IB(1200)
*
        LDA*        MCA
        ADD         =N1
        STA         BACK
        JMP         (BACK)
*
        BSS         EXTRA(6)
        END
```

```
        NAM         TRAPX
*
*       UNNEEDED ENTRY PTS PUT HERE
*
        ENT         Q8QFI,Q8QFX,Q8QFL,FOUT,FLOAT
         ENT   RFRMOT,AFRMOT,FLOTIN,RFRMIN,AFRMIN,ASCHX,BINARY
         ENT   DCHX
         ENT   EOUT
*
        EQU         Q8QFI(*),Q8QFX(*),Q8QFL(*),FOUT(*),FLOAT(*)
         EQU   RFRMOT(*),AFRMOT(*),FLOTIN(*),RFRMIN(*),AFRMIN(*),ASCHX(*)
         EQU   BINARY(*)
         EQU   DCHX(*)
         EQU   EOUT(*)
*
TRAP    NUM         0
A       JMP*        B
B       JMP*        A
        END
```

```
        NAM       CODUM
*
*       RESERVES BLOCK OF COMMON STORAGE - ELIMINATES HAND PATCHING $3F00
*       CORRELATES DUMMY PROGRAM NAMES WITH LOCATION IN SYSTEM DIRECTORY
*
        COM       DUM(255)
*
*
        ENT       D1,D2,D3,D4,D5,D6,D7,D8,D9
        ENT       D10,D11,D12,D13,D14,D15,D16
*
        EQU       D1($7E),D2($85),D3($8C),D4($93),D5($9A),D6($A1)
        EQU       D7($A8),D8($AF),D9($B6),D10($BD),D11($C4),D12($CB)
        EQU       D13($D2),D14($D9),D15($E0),D16($E7)
*
        END
```

C.2. PRINT SOURCE PROGRAM LISTING

```
      PROGRAM FINAL
C
C     READ COUNT DATA FROM DRUM,EDIT
C
C     ICEL MUST LOCATED AT $3F00
      COMMON ICEL(255)
C
      DIMENSION IB(1200)
      DIMENSION ITBL(3),ITEMP(8)
C
      RELATIVE  DRUMIO,TYPOUT,COMENT,RELESE,LINK,READ,DISPAT
C
      NCH = 400
      IADR = LINK(0) + ICEL(1)
      ILTW = ICEL(IADR+13)
      IRTW = ICEL(IADR+4)
      IF (ILTW.GT.400) GO TO 700
      IF (ILTW.LT.1) GO TO 700
      IF (IRTW.GT.400) GO TO 700
      IF (IRTW.LT.1) GO TO 700
      GO TO 750
  700 CALL COMENT(3)
      GO TO 900
C
C
  750 CONTINUE
      CALL COMENT(1)
      CALL DRUMIO(IB,2,NCH)
      ASSIGN 800 TO ICOMP
      ITBL(1) = 1
      ITBL(2) = 7
      ITBL(3) = 1502
      CALL READ($941,I1,ITBL,ICOMP,$44,ITEMP)
      CALL DISPAT
  800 CONTINUE
      I2 = ILTW
      I3 = IRTW
      CALL TYPOUT(IB,I1,I2,I3,NCH)
C
      IF (I3.NE.NCH) GO TO 900
      CALL COMENT(2)
  900 CALL RELESE(FINAL)
      GO TO 900
      END
```

```
      SUBROUTINE COMENT(NN)
C
C     PRINTS HOLLERITH ----  1-COUNT HEADING, 2-PUNCH TAPE, 3-TRY AGAIN
C
      DIMENSION ITBL(3),ITEMP(8),IBUF(23)
C
      RELATIVE  ENCODE,FWRITE,DISPAT
      RELATIVE  Q8PKUP,Q8PREP
C
  100 FORMAT(39HOCHANNEL    COUNT.1   COUNT.2   COUNT.3/)
  200 FORMAT(39HO **** PUNCH PAPER TAPE IF FINISHED    /)
  300 FORMAT(39HO CHECK THUMB WHEELS --- TRY AGAIN     /)
C
      LU = $194F
      ASSIGN 800 TO ICOMP
      GO TO (11,12,13),NN
   11 ASSIGN 100 TO IFORM
      GO TO 20
   12 ASSIGN 200 TO IFORM
      GO TO 20
   13 ASSIGN 300 TO IFORM
   20 CONTINUE
      CALL ENCODE(IBUF,IFORM,0,0)
      CALL FWRITE(LU,IBUF,20,ICOMP,$44,ITEMP)
      CALL DISPAT
  800 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE TYPOUT(IB,I1,I2,I3,NCH)
C
C     WRITES COUNTS ON LOGGING TYPEWRITER
C
C     I1        INITIAL TIME SEG TO BE PRINTED
C     I2        PRINT EVERY I2 SEGMENTS
C     I3        LAST CHANNEL TO BE PRINTED
C
      COMMON ICEL(255)
C
      EQUIVALENCE (ICEL(250),ITST7)
C
      DIMENSION IB(1200)
      DIMENSION ITBL(3),ITEMP(8)
      DIMENSION IBUF(23), LIST(4)
C
      RELATIVE  FWRITE,ENCODE,DISPAT
      RELATIVE  Q8PKUP,Q8PREP
      RELATIVE SET7,RSET7
C
C
  101 FORMAT(I5,2X,3I10)
  102 FORMAT(16H NO.OF PULSES = ,I6//)
C
      ITST7 = 0
C
C     SET UP LINE 7 INTERRUPT TRAP
      CALL SET7
C
C
C     OUTPUT COUNT DATA
C
      LU = $194F
      ASSIGN 500 TO ICOMP
      ASSIGN 101 TO IFORM
C
      DO 500 I=I1,I3,I2
      IF (ITST7.NE.0) GO TO 600
      LIST(1) = I
      LIST(2) = IB(I)
      IX = I + NCH
      LIST(3) = IB(IX)
      IY = I + NCH*2
      LIST(4) = IB(IY)
      CALL ENCODE(IBUF,IFORM,4,LIST)
      CALL FWRITE(LU,IBUF,19,ICOMP,$44,ITEMP)
      CALL DISPAT
  500 CONTINUE
C
  600 CONTINUE
C
C     RESTORE LINE 7 INTERRUPT TRAP
      CALL RSET7
```

```
C
C       OUTPUT PULSES
C
        ASSIGN 700 TO ICOMP
        ASSIGN 102 TO IFORM
        LIST(1) = IB(1200)
        CALL ENCODE(IBUF,IFORM,1,LIST)
        CALL FWRITE(LU,IBUF,12,ICOMP,$44,ITEMP)
        CALL DISPAT
  700   CONTINUE
C
  900   RETURN
        END
```

```
      SUBROUTINE DRUMIO(IB,NDO,NCH)
C
C     READ OR WRITE COUNT DATA ON DRUM  NDO=1-WRITE,2-READ
C
      DIMENSION IB(1200)
      DIMENSION ITBL(3),ITEMP(8)
C
      RELATIVE  WRITE,READ,DISPAT
      RELATIVE  Q8PKUP,Q8PREP
C
      LU = $941
      ITBL(1) = NCH
      ITBL(2) = 7
C
      ASSIGN 500 TO ICOMP
      DO 500 K=1,3
      L = (K-1)*NCH + 1
      ITBL(3) = L
      GO TO (1,2),NDO
    1 CALL WRITE(LU,IB(L),ITBL,ICOMP,$44,ITEMP)
      GO TO 3
    2 CALL READ (LU,IB(L),ITBL,ICOMP,$44,ITEMP)
    3 CALL DISPAT
  500 CONTINUE
      RETURN
      END
```

```
        NAM         TRAPX
*
*       UNNEEDED ENTRY PTS PUT HERE
*
        ENT         Q8QFI,Q8QFX,Q8QFL,FOUT,FLOAT
         ENT   RFRMOT,AFRMOT,FLOTIN,RFRMIN,AFRMIN,ASCHX,BINARY
         ENT   DCHX
         ENT   EOUT
*
        EQU         Q8QFI(*),Q8QFX(*),Q8QFL(*),FOUT(*),FLOAT(*)
         EQU   RFRMOT(*),AFRMOT(*),FLOTIN(*),RFRMIN(*),AFRMIN(*),ASCHX(*)
         EQU   BINARY(*)
         EQU   DCHX(*)
         EQU   EOUT(*)
*
TRAP    NUM         0
A       JMP*        B
B       JMP*        A
        END
```

```
        NAM         SET7                SETS UP LINE 7 INTERRUPT TRAP
*
        ENT         SET7
*
        EXT         TRAP7
*
LOC7    JMP         TRAP7
SET7    NUM         0
        IIN
        LDA         =N$1C02             JMP    (+2)
        STA+        $11D
        LDA         LOC7+1              LOCATION OF INTERRUPT PROCESSOR 7
        STA+        $11F
        EIN
        JMP*        (SET7)
        END
```

```
       NAM       RSET7                RESETS INTERRUPT TRAP 7
*
       ENT       RSET7
*
RSET7  NUM       0                    ENTRY POINT
       IIN
       LDA       =N$54FE
       STA+      $11D
       LDA       =N$675
       STA+      $11F
       EIN
       JMP*      (RSET7)
       END
```

```
       NAM       TRAP7                INTERRUPT PROCESSOR FOR LINE 7
*
       COM       DUM(255)
*
       EQU       ITST7(DUM+249)
*
       ENT       TRAP7
*
TRAP7  NUM       0          ENTRY POINT FOR LINE 7 PROCESSOR
       ENA       -1         PUT -1 IN LOC TEST7 IF ENTER BUTTON DEPRESSED
       STA       ITST7
       LDQ       =N$400
       INP       -1
       LDQ       =N$9000
       INP       -1
       INQ       1
       INP       -1
       EXI       $1C        EXIT FROM INTERRUPT PROCESSOR
       END
```

```
        NAM       CODUM
*
*       RESERVES BLOCK OF COMMON STORAGE - ELIMINATES HAND PATCHING $3F00
*       CORRELATES DUMMY PROGRAM NAMES WITH LOCATION IN SYSTEM DIRECTORY
*
        COM       DUM(255)
*
*
        ENT       D1,D2,D3,D4,D5,D6,D7,D8,D9
        ENT       D10,D11,D12,D13,D14,D15,D16
*
        EQU       D1($7E),D2($85),D3($8C),D4($93),D5($9A),D6($A1)
        EQU       D7($A8),D8($AF),D9($B6),D10($BD),D11($C4),D12($CB)
        EQU       D13($D2),D14($D9),D15($E0),D16($E7)
*
        END
```

C.3. ALCALC SOURCE PROGRAM LISTING

```
      PROGRAM ALCALC
C
C     CORRECTS FOR COUNTING LOSSES,OBTAINS BACKGROUND FROM PULSE SOURCE
C     EXPERIMENT DATA, THEN COMPUTES PROMPT DECAY CONSTANT
C
C
C     WIDTH      CHANNEL WIDTH (MICROSECONDS)
C     FREQ       PULSE RATE (PULSES/SEC)
C     PULSES     TOTAL PULSE RECORDED FOR EXPERIMENT
C     TAU        CHANNEL DEAD TIME
C     ITAU       CHANNEL DEAD TIME IN NANOSECONDS
C     C(I)       COUNT DATA FOR A CHANNEL
C     IC(I)      COUNT DATA (INTEGER) UNPROCESSED
C     W(I)       WEIGHT APPLIED TO COUNT DATA
C     ILTW       0 - USE PREVIOUSLY OBTAINED CHANNEL RANGES
C                1,2,3 - DETERMINE RANGE FROM THIS DETECTORS DATA
C     CRAW       RAW COUNTS
C     NP1,2      FIRST AND LAST PROMPT DECAY CHANNEL
C     NFRST      FIRST BACKGROUND CHANNEL
C     NLAST      LAST  BACKGROUND CHANNEL
C     NGPS       NUMBER OF GROUPS OF CHANNELS FOR ALPHA CALCULATION
C     NGPW       NUMBER OF CHANNELS/CHANNEL GROUP FOR ALPHA CALCULATION
C     NERROR     ERROR INDICATOR  0 - APPARENTLY OK,EDIT RESULTS
C                                 1 - TOO MANY BAD DATA CHANNELS
C                                 2 - TOO MANY BACKGROUND CHANNELS
C                                 3 - CHECK THUMBWHEELS
C                                 4 -
C                                 5 -
C     KBG        NUMBER OF BACKGROUND CHANNEL GROUPS
C     NBG1,2     FIRST AND LAST CHANNEL IN BACKGROUND GROUP
C     BG         AVERAGE B.G. COUNT FOR A BACKGROUND GROUP
C     BGBASE     RUNNING AVERAGE BACKGROUND COUNT
C     BGBAR(J)   AVERAGE BACKGROUND FOR JTH DETECTOR
C     COE(N)     POLYNOMIAL FIT COEFFICIENTS
C     NDET       NOT USED
C     ALPHA(I,J) ALPHA FOR DETECTOR I,CHANNEL GRP J
C     DELAL(I,J) ALPHA UNCERTAINTY FOR DETECTOR I, CHANNEL GRP J
C     ALP(I)     ALPHA FOR ALL OF PROMPT RANGE
C     DEL(I)     ALPHA UNCERTAINTY FOR ALL OF PROMPT RANGE
C     T(I)       TIME AFTER PULSE FOR CHANNEL I (MID-CHANNEL)
C     RHO(J)     REACTIVITY
C     DOLLAR(J)  DOLLARS OF REACTIVITY
C     BETA       DELAYED NEUTRON FRACTION
C     XLAM       NEUTRON GENERATION TIME
C
C
C
      DIMENSION C(25),IC(25),W(25),T(25),COE(5),ALPHA(8,3),BGBAR(3)
      DIMENSION DELAL(8,3),ALP(3),DEL(3)
      DIMENSION NRNG(20) ,IDMP(134) ,ITEMP(4) ,ICN(2)
      DIMENSION CN(2)    ,RHO(3)    ,DOLLAR(3)          ,XKEFF(3)
C
      EQUIVALENCE (NRNG(1),NP1)   ,(NRNG(6),ITAU)   ,(NRNG(11),NGPW)
```

```
      EQUIVALENCE (NRNG(2),NP2)    ,(NRNG(7),IWIDTH) ,(NRNG(12),NERROR)
      EQUIVALENCE (NRNG(3),NFRST) ,(NRNG(8),ICN(1)) ,(NRNG(13),ILTW)
      EQUIVALENCE (NRNG(4),NLAST) ,(NRNG(9),ICN(2)) ,(NRNG(14),IPULSE)
      EQUIVALENCE (NRNG(5),NGPS)  ,(NRNG(10),NDET)  ,(NRNG(15),IRTW)
      EQUIVALENCE (NRNG(16),NPASS),(NRNG(17),IMIN)
      EQUIVALENCE (NRNG(18),IBETA),(NRNG(19),LAMDA)
      EQUIVALENCE (IDMP(1),ALPHA(1,1))
      EQUIVALENCE (IDMP(49),DELAL(1,1))
      EQUIVALENCE (IDMP(97),ALP(1))
      EQUIVALENCE (IDMP(103),DEL(1))
      EQUIVALENCE (IDMP(109),BGBAR(1))
      EQUIVALENCE (IDMP(115),WIDTH)
      EQUIVALENCE (IDMP(117),RHO(1))
      EQUIVALENCE (IDMP(123),DOLLAR(1))
      EQUIVALENCE (IDMP(129),XKEFF(1))
C
      COMMON ICEL(255)
C
      RELATIVE LINK,DRMDAT,TCHAN,CWDATA,SCHEDL,RELESE,FIT2
      RELATIVE ALOG,EXP,ABS,SQRT,FLOAT
      RELATIVE FLOT,Q3QFLT,Q8QF2I
      RELATIVE TEST
C
      EXTERNAL D6
C
      IADR = LINK(0) + ICEL(1)
C
C     READ TABLE OF MANUAL INPUT ITEMS
C
      CALL DRMDAT(NRNG,7,1504,20,2)
C
      IRTW = ICEL(IADR+4)
      ILTW = ICEL(IADR+13)
      L1 = 1 + 16*(ILTW-1)
      L2 = L1 + 15
      IF (ILTW.GT.3) GO TO 1003
      NERROR = 0
                                                    CALL TEST(1)
C
C     READ NUMBER OF PULSES
C
      CALL DRMDAT(IPULSE,7,1200,1,2)
      PULSES= FLOAT(IPULSE)
C
      TAU = FLOAT(ITAU)*1.0E-9
      WIDTH = FLOAT(NRNG(7))*1.0E-6
      CN(1) = FLOAT(ICN(1))*1.0E-3
      CN(2) = FLOAT(ICN(2))*1.0E-3
      BETA  = FLOAT(IBETA)*1.0E-6
      XLAM  = FLOAT(LAMDA)*1.0E-6
C
C     CORRECT FOR COUNTING LOSSES AND COMPUTE WEIGHTS
C
      CORMIN = FLOAT(IMIN)*1.0E-3
```

```
      FACT = TAU/(PULSES*(WIDTH+.5*TAU))
      TESFAC = CORMIN/FACT
C
      DO 100 J=1,48
                                                      CALL TEST(2)
      ILOC = 25*J - 24
      CALL DRMDAT(IC,7,ILOC,25,2)
      IF (J.EQ.48) IC(25)=0
C
      DO 50 I=1,25
      IF (IC(I).LT.1) GO TO 30
      CRAW = FLOAT(IC(I))
      IF (CRAW.LT.TESFAC) GO TO 27
      C(I) = CRAW/(1.0-CRAW*FACT)
      IF (C(I).LT.1.0) GO TO 30
      SIG = SQRT(CRAW)/C(I)
      W(I) = 1.0/(SIG**2)
      GO TO 33
   27 CONTINUE
      C(I) = CRAW
      W(I) = CRAW
      GO TO 33
   30 CONTINUE
      C(I) = 0.0
      W(I) = 0.0
   33 CONTINUE
C
C     FIND LAST CHANNEL WITH COUNTS RECORDED
C
      IF (ILTW.EQ.0) GO TO 40
      IF (J.LT.L1) GO TO 40
      IF (J.GT.L2) GO TO 40
      II = (J-1)*25 + I - 400*(ILTW-1)
      IF (II.EQ.400) GO TO 40
      IF (IC(I).NE.0) NLAST=II
   40 CONTINUE
   50 CONTINUE
C
      ICLOC = 50*J-49 + 3000
      IWLOC = 50*J-49 + 5400
      CALL DRMDAT(C,7,ICLOC,50,1)
      CALL DRMDAT(W,7,IWLOC,50,1)
  100 CONTINUE
C
      IF (NLAST.LT.10) GO TO 1004
      IF (NPASS.EQ.0) GO TO 207
C
C     THROW OUT OBVIOUSLY BAD DATA POINTS
C
      NBAD = 0
      DO 200 J=1,48
                                                      CALL TEST(3)
      ICLOC = 50*J-49 + 3000
      IWLOC = 50*J-49 + 5400
```

```
      CALL DRMDAT(C,7,ICLOC,50,2)
      CALL DRMDAT(W,7,IWLOC,50,2)
      N = 0
      IF (J.GT.16) N=400
      IF (J.GT.32) N=800
      N1 = 25*J - 24 - N
      N2 = N1 + 24
      CALL TCHAN(T,N1,N2,WIDTH)
C
      DO 130 II=1,25
      IF (C(II).LT.1.0) GO TO 120
      C(II) = ALOG(C(II))
      IF (C(II).GT.10.38) GO TO 120
      GO TO 130
  120 C(II) = 0.0
      W(II) =0.0
  130 CONTINUE
C
      CS = CN(1)
      DO 160 I=1,2
      TTTT = CS
      IF (CS.LE.0.0) GO TO 160
      MM = 25
      IF (J.EQ.48) MM=24
      CALL FIT2(C,T,W,MM,COE,D)
C
      DO 150 II=1,25
      ARG = COE(1)+COE(2)*T(II)
      IF (ARG.GT.10.38) GO TO 135
      IF (ARG.LT.0.0) GO TO 135
      IF (ABS(ARG-C(II)).LT.TTTT) GO TO 140
  135 CONTINUE
C
C     WRITE BAD CHANNEL DATA ON DRUM - W=0.0,C=CFIT
C
      W(II) = 0.0
      CFIT = EXP(ARG)
      IADD = 2*(II-1)
      ICL = ICLOC + IADD
      IWL = IWLOC + IADD
      CALL DRMDAT(CFIT,7,ICL,2,1)
      CALL DRMDAT(W(II),7,IWL,2,1)
C
C     DETERMINE NUMBER OF CHANNELS WITH BAD DATA FOR SPECIFIED DETECTOR
C                                   - SKIP DATA FOR OTHER DETECTORS
C
      IF (J.LT.L1) GO TO 140
      IF (J.GT.L2) GO TO 140
      NT = N1 + II - 1
      IF (NT.GT.NLAST) GO TO 140
      IF (I.EQ.1) GO TO 140
      NBAD = NBAD + 1
  140 CONTINUE
  150 CONTINUE
```

```
      CS = CN(2)
  160 CONTINUE
  200 CONTINUE
C
C     IF TOO MANY BAD DATA VALUES ARE FOUND,STOP CALCULATION
C
      XBAD = FLOAT(NBAD)/FLOAT(NLAST)
      IF (XBAD.GT.0.30) GO TO 1001
C
  207 CONTINUE
C
C     OBTAIN B.G. RANGE FOR SPECIFIED DETECTOR
C                 (AVERAGE B.G. IN GROUPS OF 25 CHANNELS BACK TO PROMPT)
C
      IF (ILTW.EQ.0) GO TO 300
      BGBASE = 0.0
      KBG = 0
C
  220 KBG = KBG + 1
                                                      CALL TEST(4)
      NBG2 = NLAST - 25*KBG
      NBG1 = NBG2 - 24
      IF (NBG1.LT.1) GO TO 1002
      CALL CWDATA(C,NBG1,NBG2,ILTW,2)
      BG = 0.0
      DO 240 I=1,25
  240 BG = BG +C(I)*.04
      IF (KBG.EQ.1) GO TO 250
      IF (ABS(BG-BGBASE).GT.2.0*SQRT(BGBASE)) GO TO 270
  250 CONTINUE
      XBG = FLOAT(KBG)
      BGBASE = (BG+(XBG-1.0)*BGBASE)/XBG
      GO TO 220
C
  270 NFRST = NBG1 + 25
      GO TO 330
C
C     USE B.G. CHANNEL RANGES FROM DRUM
C
  300 CONTINUE
  330 CONTINUE
C
C     AVERAGE BACKGROUND FOR ALL CHANNELS - NFRST,NLAST
C
      DO 400 I=1,3
                                                      CALL TEST(5)
      BGBAR(I) = 0.0
      DO 380 KBG=NFRST,NLAST,25
      NEND = KBG + 24
      IF (NEND.GE.NLAST) NEND=NLAST
      CALL CWDATA(C,KBG,NEND,I,2)
      NSTOP = NEND - KBG + 1
      DO 370 J=1,NSTOP
  370 BGBAR(I) = BGBAR(I) + C(J)
```

```
C
  380 CONTINUE
      DIV = NLAST - NFRST + 1
      BGBAR(I) = BGBAR(I)/DIV
  400 CONTINUE
C
C
C     FIND FIRST CHANNEL FOR PROMPT DECAY CURVE (FIND MAX COUNTS)
C
      IF (ILTW.EQ.0) GO TO 600
      NBIG = 1
      CBIG = 0.0
      DO 480 NP=1,NFRST,25
                                                    CALL TEST(6)
      NEND = NP+24
      IF (NEND.GT.NFRST) NEND=NFRST
      CALL CWDATA(C,NP,NEND,ILTW,2)
      CALL CWDATA(W,NP,NEND,ILTW,4)
      NSTOP = NEND-NP+1
      DO 470 J=1,NSTOP
      IF (C(J).LE.CBIG) GO TO 470
      IF (W(J).LE.0.0) GO TO 470
      NBIG = J + NP
      CBIG = C(J)
  470 CONTINUE
  480 CONTINUE
      NP1 = NBIG
C
C     FIND LAST CHANNEL FOR PROMPT DECAY CURVE
C               (LINEAR FIT - FIND VALUE LESS THAN 3.0 X BG)
C
      DO 580 NP=NBIG,NFRST,20
                                                    CALL TEST(7)
      NEND = NP+19
      CALL CWDATA(W,NP,NEND,ILTW,4)
      CALL CWDATA(C,NP,NEND,ILTW,2)
      DO 540 J=1,20
  540 C(J) = ALOG(C(J))
      CALL TCHAN(T,NP,NEND,WIDTH)
      CALL FIT2(C,T,W,20,COE,D)
      DO 550 J=1,20
      NP2 = J+NP-1
      ARG = COE(1) + COE(2)*T(J)
      IF (ARG.GT.10.38) GO TO 550
      CFIT = EXP(ARG)
      IF (CFIT.LT.3.0*BGBAR(ILTW)) GO TO 590
  550 CONTINUE
  580 CONTINUE
  590 CONTINUE
  600 CONTINUE
C
C     DIVIDE RANGE INTO SMALLER GROUPS OF CHANNELS FOR ALPHA CALCULATION
C
      NPRNG = NP2-NP1+1
```

```
      NGPW = NPRNG/NGPS
      IF (NGPW.GT.25) NGPW=25
      IF (NGPW.GE.10) GO TO 620
      NGPW = 10
      NGPS = NPRNG/NGPW
  620 CONTINUE
      IF (NGPS.GT.8) NGPS=8
      IF (NGPS*NGPW.LT.NPRNG) NP1=NP2-NGPS*NGPW+1
C
C     WRITE RANGES ETC. BACK INTO MANUAL INPUT TABLE ON DRUM
C
      CALL DRMDAT(NRNG,7,1504,20,1)
C
C     REMOVE BACKGROUND FROM COUNTS,TAKE LOG,AND FIT TO GET ALPHA
C
      DO 700 J=1,3
                                                  CALL TEST(8)
      IG = 0
      DO 650 I=NP1,NP2,NGPW
      IG = IG + 1
      IE = I+NGPW-1
      CALL CWDATA(W,I,IE,J,4)
      CALL CWDATA(C,I,IE,J,2)
      CALL TCHAN(T,I,IE,WIDTH)
      DO 640 II=1,NGPW
      IF (C(II).LE.BGBAR(J)) GO TO 633
      GO TO 635
  633 W(II) = 0.0
      C(II) = 0.0
      GO TO 640
  635 CONTINUE
      C(II) = ALOG(C(II)-BGBAR(J))
  640 CONTINUE
C
      CALL FIT2(C,T,W,NGPW,COE,D)
C
      ALPHA(IG,J) = COE(2)
C
C     NOW GET ALPHA UNCERTAINTY FOR EACH CHANNEL GROUP
C
      XGPW = NGPW - 2
      SUM = 0.0
      SUMW = 0.0
      DO 645 II=1,NGPW
      SUMW = SUMW + W(II)
      SUM = SUM + (COE(1)+COE(2)*T(II)-C(II))**2*W(II)
  645 CONTINUE
      DELAL(IG,J) = 0.0
      IF (D.EQ.0.0) GO TO 650
      SUM = SUM*SUMW/(XGPW*D)
      DELAL(IG,J) = SQRT(SUM)
  650 CONTINUE
C
C     COMPUTE ALPHA AND UNCERTAINTY FOR ENTIRE RANGE
```

```
C
      DELS = 0.0
      RHO(J) = 0.0
      DOLLAR(J) = 0.0
      XKEFF(J) = 0.0
      ALP(J) = 0.0
      DEL(J) = 0.0
      WTS = 0.0
      DO 680 I=1,NGPS
      IF (DELAL(I,J).EQ.0.0) GO TO 680
      WT = (1.0/DELAL(I,J))**2
      WTS = WTS + WT
      ALP(J) = ALP(J) + WT*ALPHA(I,J)
  680 CONTINUE
      IF (WTS.EQ.0.0) GO TO 690
      ALP(J) = ALP(J)/WTS
      DEL(J) = SQRT(1.0/WTS)
C
C     COMPUTE REACTIVITY IMPLIED BY ALPHAS
C
      RHO(J) = BETA + XLAM*ALP(J)
      DOLLAR(J) = RHO(J)/BETA
      XKEFF(J) = 1.0/(1.0-RHO(J))
  690 CONTINUE
  700 CONTINUE
  900 CONTINUE
C
C     WRITE DATA INTO TEMPORARY DRUM STORAGE READY FOR PRINTING
C
 1000 CONTINUE
C
      CALL DRMDAT(IDMP,7,7801,134,1)
      CALL DRMDAT(NRNG,7,7935,20,1)
C
C
C     DE IS ALPRN
      CALL SCHEDL(DS,$104,0,ITEMP)
      CALL RELESE(ALCALC)
      GO TO 1000
C
 1001 NERROR = 1
      GO TO 1000
 1002 NERROR = 2
      GO TO 1000
 1003 NERROR = 3
      GO TO 1000
 1004 NERROR = 4
      GO TO 1000
      END
```

```
      SUBROUTINE CWDATA(C,NCH1,NCH2,NDET,NCW)
C
C     WRITES OR READSCOUNTS OR WEIGHTS ON/FROM DRUM
C
C     C          DATA
C     NCH1       FIRST CHANNEL
C     NCH2       LAST CHANNEL
C     NDET       DETECTOR
C     NCW        1-WRITE COUNTS, 2-READ COUNNTS, 3-WRITE WEIGHTS, 4-READ  WEIGHTS
C
      RELATIVE  Q8PKUP,Q8PREP,DRMDAT
C
      GO TO (100,100,200,200),NCW
C
  100 ILOC = 3000
      GO TO 300
  200 ILOC = 5400
  300 ILOC = ILOC + (NDET-1)*800 + NCH1*2 - 1
      IWORDS =(NCH2-NCH1+1)*2
      NDO = 1
      GO TO (500,400,500,400),NCW
  400 NDO = 2
  500 CALL DRMDAT(C,7,ILOC,IWORDS,NDO)
      RETURN
      END
```

```
      SUBROUTINE DRMDAT(IZ,IPG,ILOC,IWORDS,NDO)
C
C     READS OR WRITES BLOCKS OF DATA FROM/TO DRUM STORAGE
C
C     IZ          DATA BLOCK
C     IPG         DRUM PAGE
C     ILOC        DRUM POSITION OF FIRST WORD OF DATA BLOCK
C     IWORDS      NUMBER OF WORDS TO BE TRANSFERED
C     NDO         1-WRITE,  2-READ
C
      DIMENSION IZ(50),ITBL(3),ITEMP(8)
C
      RELATIVE  WRITE,READ,DISPAT
      RELATIVE  Q8PKUP,Q8PREP
C
      LU = $941
      ITBL(1) = IWORDS
      ITBL(2) = IPG
      ITBL(3) = ILOC
      ASSIGN 500 TO ICOMP
      GO TO (1,2),NDO
    1 CALL WRITE(LU,IZ,ITBL,ICOMP,$44,ITEMP)
      GO TO 3
    2 CALL READ(LU,IZ,ITBL,ICOMP,$44,ITEMP)
    3 CALL DISPAT
  500 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE FIT2(Y,X,W,N,COE,D)
C
C     PERFORM FIRST ORDER POLYNOMIAL FIT
C
C     COE(I)    POLYNOMIAL FIT COEFF
      DIMENSION Y(25),X(25),W(25),XK(5),XC(5),COE(5)
C
      RELATIVE  FLOT,Q8PKUP,Q8PREP,Q8QF2I
C
      DO 100 I=1,3
      XK(I) = 0.0
      XC(I) = 0.0
      COE(I) = 0.0
  100 CONTINUE
C
C
      DO 200 I=1,N
      XK(1) = XK(1) + W(I)
      XK(2) = XK(2) + W(I)*X(I)
      XK(3) = XK(3) + W(I)*X(I)**2
      XC(1) = XC(1) + W(I)*Y(I)
      XC(2) = XC(2) + W(I)*X(I)*Y(I)
  200 CONTINUE
C
      D      =XK(1)*XK(3) - XK(2)*XK(2)
      IF (D.EQ.0.0) GO TO 600
      COE(1) = (XC(1)*XK(3) - XK(2)*XC(2))/D
      COE(2) =(XK(1)*XC(2) - XC(1)*XK(2))/D
      COE(3) = 0.0
  600 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE TCHAN(T,N1,N2,WIDTH)
C
C     T(I)       CHANNEL POSITION IN TIME (NOMINAL)
C     N1         FIRST CHANNEL NUMBER
C     N2         LAST CHANNEL NUMBER
C     WIDTH      CHANNEL WIDTH
C
C     COMPUTE TIME FOR SET OF CHANNELS
C
      DIMENSION T(25)
C
      RELATIVE  FLOT,Q8PKUP,Q8PREP,FLOAT
C
      NEND = N2-N1+1
      DO 100 I=1,NEND
      K = N1+I-1
      T(I) = (FLOAT(K)-.5)*WIDTH
  100 CONTINUE
      RETURN
      END
```

```
SUBROUTINE TEST(K)
RELATIVE Q8PKUP,Q8PREP,DRMDAT
ILOC = 1535
CALL DRMDAT(K,7,ILOC,1,1)
RETURN
END
```

```
        NAM         TRAPY
*
*       UNNEEDED ENTRY PTS PUT HERE
*
        ENT         Q8QFI,Q8QFX,Q8QFL,FOUT
         ENT   RFRMOT,AFRMOT,FLOTIN,RFRMIN,AFRMIN,ASCHX,BINARY
         ENT   DCHX
         ENT   EOUT
*
        EQU         Q8QFI(*),Q8QFX(*),Q8QFL(*),FOUT(*)
         EQU   RFRMOT(*),AFRMOT(*),FLOTIN(*),RFRMIN(*),AFRMIN(*),ASCHX(*)
         EQU   BINARY(*)
         EQU   DCHX(*)
         EQU   EOUT(*)
*
TRAP    NUM         0
A       JMP*        B
B       JMP*        A
        JMP*        A
        END
```

```
        NAM       CODUM
*
*       RESERVES BLOCK OF COMMON STORAGE - ELIMINATES HAND PATCHING $3FDD
*       CORRELATES DUMMY PROGRAM NAMES WITH LOCATION IN SYSTEM DIRECTORY
*
        COM       DUM(255)
*
*
        ENT       D1,D2,D3,D4,D5,D6,D7,D8,D9
        ENT       D10,D11,D12,D13,D14,D15,D16
*
        EQU       D1($7E),D2($85),D3($8C),D4($93),D5($9A),D6($A1)
        EQU       D7($A8),D8($AF),D9($B6),D10($BD),D11($C4),D12($CB)
        EQU       D13($D2),D14($D9),D15($E0),D16($E7)
*
        END
```

C.4. ALPRN SOURCE PROGRAM LISTING

```
      PROGRAM ALPRN
C
C     EDITS RESULTS OF ALPHA CALCULATION
C
      DIMENSION ITBL(3),ITEMP(3),IBUF(40)
      DIMENSION LIST(40),FLIST(10),GLIST(10),IBG(3)
      DIMENSION ALPHA(3,3),BGBAR(3),IDMP(134),NRNG(20)
      DIMENSION DELAL(8,3),ALP(3),DEL(3)
      DIMENSION ICN(2),RHO(3),DOLLAR(3),XKEFF(3)
C
      EQUIVALENCE (LIST(2),FLIST(1)),(LIST(1),GLIST(1))
      EQUIVALENCE (NRNG(1),NP1)   ,(NRNG(6),ITAU)   ,(NRNG(11),NGPW)
      EQUIVALENCE (NRNG(2),NP2)   ,(NRNG(7),IWIDTH) ,(NRNG(12),NERROR)
      EQUIVALENCE (NRNG(3),NFRST) ,(NRNG(8),ICN(1)) ,(NRNG(13),ILTW)
      EQUIVALENCE (NRNG(4),NLAST) ,(NRNG(9),ICN(2)) ,(NRNG(14),IPULSE)
      EQUIVALENCE (NRNG(5),NGPS)  ,(NRNG(10),NDET)  ,(NRNG(15),IRTW)
      EQUIVALENCE (NRNG(16),NPASS),(NRNG(17),IMIN)
      EQUIVALENCE (NRNG(18),IBETA),(NRNG(19),LAMDA)
      EQUIVALENCE (IDMP(1),ALPHA(1,1))
      EQUIVALENCE (IDMP(49),DELAL(1,1))
      EQUIVALENCE (IDMP(97),ALP(1))
      EQUIVALENCE (IDMP(103),DEL(1))
      EQUIVALENCE (IDMP(109),BGBAR(1))
      EQUIVALENCE (IDMP(115),WIDTH)
      EQUIVALENCE (IDMP(117),RHO(1))
      EQUIVALENCE (IDMP(123),DOLLAR(1))
      EQUIVALENCE (IDMP(129),XKEFF(1))
C
      RELATIVE DRMDAT,ENCODE,FWRITE,DISPAT,RELESE
      RELATIVE FLOT,IFIX
C
C
  101 FORMAT(22H TOO MANY BAD CHANNELS)
  102 FORMAT(20H TOO MUCH BACKGROUND /)
  103 FORMAT(26H CHECK THUMBWHEEL SETTING /)
  104 FORMAT(10H BAD DATA //)
  105 FORMAT(10H TROUBLE  //)
  106 FORMAT(1X///72H PROMPT CHANNELS   GRP WIDTH     B.G. CHANNELS
     1   PULSES      FLAG  /)
  107 FORMAT(I5,I9,I10,I13,I8,I15,I8//)
  108 FORMAT( 9H DETECTOR,10X,1H1,22X,1H2,22X,1H3/)
  109 FORMAT(11H B.G.COUNTS,I10,15X,I8,15X,I8/)
  110 FORMAT(73H GRP       ALPHA      ERROR       ALPHA      ERROR
     1 ALPHA      ERROR  /)
  111 FORMAT(I3,1X,3(E13.3,E10.2))
  112 FORMAT( 4H AVG,3(E13.3,E10.2)///)
  113 FORMAT( 4H RHO,3(E13.3,10X))
  114 FORMAT( 4H  $ ,3(E13.3,10X))
  115 FORMAT( 4H  K ,3(E13.3,10X))
  116 FORMAT(///5X,5HLAMDA,6X,4HBETA,5X,5HDELTA,7X,3HTAU,4X,6HCORREX,3X,
     17HSMOOTH1,3X,7HSMOOTH2)
  117 FORMAT(7I10)
C
```

```
C     READ COMPUTED DATA FROM DRUM
C
      CALL DRMDAT(IDMP,7,7301,134,2)
      CALL DRMDAT(NRNG,7,7935,20,2)
C
      LU = $194F
C
      IF (NERROR.EQ.0) GO TO 50
      IF (NERROR.GT.5)  GO TO 1005
      GO TO (1001,1002,1003,1004,1005),NERROR
C
C     EDIT INTEGER DATA
C
   50 CONTINUE
      ASSIGN 206 TO ICOMP
      ASSIGN 106 TO IFORM
      CALL ENCODE(IBUF,IFORM,0,0)
      CALL FWRITE(LU,IBUF,38,ICOMP,$44,ITEMP)
      CALL DISPAT
  206 CONTINUE
C
      ASSIGN 207 TO ICOMP
      ASSIGN 107 TO IFORM
      LIST(1) = NP1
      LIST(2) = NP2
      LIST(3) = NGPW
      LIST(4) = NFRST
      LIST(5) = NLAST
      LIST(6) = IPULSE
      LIST(7) = ILTW
      CALL ENCODE(IBUF,IFORM,7,LIST)
      CALL FWRITE(LU,IBUF,35,ICOMP,$44,ITEMP)
      CALL DISPAT
  207 CONTINUE
C
C     EDIT BACKGROUND COUNTS
C
      ASSIGN 208 TO ICOMP
      ASSIGN 108 TO IFORM
      CALL ENCODE(IBUF,IFORM,0,0)
      CALL FWRITE(LU,IBUF,33,ICOMP,$44,ITEMP)
      CALL DISPAT
  208 CONTINUE
C
      ASSIGN 209 TO ICOMP
      ASSIGN 109 TO IFORM
      DO 309 I=1,3
  309 IBG(I) = IFIX(BGBAR(I))
      CALL ENCODE(IBUF,IFORM,3,IBG)
      CALL FWRITE(LU,IBUF,34,ICOMP,$44,ITEMP)
      CALL DISPAT
  209 CONTINUE
C
C     EDIT ALPHA,ERROR BY GROUP FOR 3 DETECTORS
```

```
C
      ASSIGN 210 TO ICOMP
      ASSIGN 110 TO IFORM
      CALL ENCODE(IBUF,IFORM,0,0)
      CALL FWRITE(LU,IBUF,37,ICOMP,$44,ITEMP)
      CALL DISPAT
  210 CONTINUE
C
      IF (NGPS.GT.8) GO TO 1005
      DO 311 I=1,NGPS
      ASSIGN 211 TO ICOMP
      ASSIGN 111 TO IFORM
      LIST(1) = I
      DO 411 K=1,3
      N = 2*K - 1
      FLIST(N) = ALPHA(I,K)
      FLIST(N+1) = DELAL(I,K)
  411 CONTINUE
      CALL ENCODE(IBUF,IFORM,7,LIST)
      CALL FWRITE(LU,IBUF,37,ICOMP,$44,ITEMP)
      CALL DISPAT
  211 CONTINUE
  311 CONTINUE
C
C     EDIT ALPHA,ERROR (AVERAGE) FOR 3 DETECTORS
C
      ASSIGN 212 TO ICOMP
      ASSIGN 112 TO IFORM
      DO 312 I=1,3
      N = 2*I -1
      GLIST(N) = ALP(I)
      GLIST(N+1)= DEL(I)
  312 CONTINUE
      CALL ENCODE(IBUF,IFORM,6,LIST)
      CALL FWRITE(LU,IBUF,38,ICOMP,$44,ITEMP)
      CALL DISPAT
  212 CONTINUE
C
C     PRINT REACTIVITY,DOLLARS,AND K-EFF IMPLIED BY ALPHAS
C
      DO 350 J=1,3
      ASSIGN 340 TO ICOMP
      GO TO (223,224,225),J
  223 ASSIGN 113 TO IFORM
      GO TO 230
  224 ASSIGN 114 TO IFORM
      GO TO 230
  225 ASSIGN 115 TO IFORM
  230 CONTINUE
      DO 320 I=1,3
      GO TO (233,234,235),J
  233 GLIST(I) = RHO(I)
      GO TO 320
  234 GLIST(I) = DOLLAR(I)
```

```
      GO TO 320
  235 GLIST(I) = XKEFF(I)
  320 CONTINUE
      CALL ENCODE(IBUF,IFORM,3,GLIST)
      CALL FWRITE(LU,IBUF,35,ICOMP,$44,ITEMP)
      CALL DISPAT
  340 CONTINUE
  350 CONTINUE
C
C     PRINT APPLICABLE DATA FROM MANUAL INPUT TABLE
C
      ASSIGN 260 TO ICOMP
      ASSIGN 116 TO IFORM
      CALL ENCODE(IBUF,IFORM,0,0)
      CALL FWRITE(LU,IBUF,35,ICOMP,$44,ITEMP)
      CALL DISPAT
  260 CONTINUE
      ASSIGN 270 TO ICOMP
      ASSIGN 117 TO IFORM
      DO 280 I=1,2
      GO TO (263,266),I
  263 LIST(1) = LAMDA
      LIST(2) = IBETA
      LIST(3) = IWIDTH
      LIST(4) = ITAU
      LIST(5) = IMIN
      LIST(6) = ICN(1)
      LIST(7) = ICN(2)
      GO TO 269
  266 LIST(1) = -6
      LIST(2) = -6
      LIST(3) = -6
      LIST(4) = -9
      LIST(5) = -3
      LIST(6) = -3
      LIST(7) = -3
  269 CONTINUE
      CALL ENCODE(IBUF,IFORM,7,LIST)
      CALL FWRITE(LU,IBUF,35,ICOMP,$44,ITEMP)
      CALL DISPAT
  270 CONTINUE
  280 CONTINUE
C
C
  900 CONTINUE
 1000 CALL RELESE(ALPRN)
C
C     EDIT ERROR COMMENT
C
 1001 ASSIGN 101 TO IFORM
      JJ = 11
      GO TO 90
 1002 ASSIGN 102 TO IFORM
      JJ = 11
```

```
      GO TO 90
 1003 ASSIGN 103 TO IFORM
      JJ = 14
      GO TO 90
 1004 ASSIGN 104 TO IFORM
      JJ = 6
      GO TO 90
 1005 ASSIGN 105 TO IFORM
      JJ = 6
C
   90 CONTINUE
      ASSIGN 99 TO ICOMP
      CALL ENCODE(IBUF,IFORM,0,C)
      CALL FWRITE(LU,IBUF,JJ,ICOMP,$44,ITEMP)
      CALL DISPAT
   99 CONTINUE
      GO TO 1000
      END
```

```
      SUBROUTINE DRMDAT(IZ,IPG,ILOC,IWORDS,NDO)
C
C     READS OR WRITES BLOCKS OF DATA FROM/TO DRUM STORAGE
C
C     IZ          DATA BLOCK
C     IPG         DRUM PAGE
C     ILOC        DRUM POSITION OF FIRST WORD OF DATA BLOCK
C     IWORDS      NUMBER OF WORDS TO BE TRANSFERED
C     NDO         1-WRITE,  2-READ
C
      DIMENSION IZ(50),ITBL(3),ITEMP(8)
C
      RELATIVE  WRITE,READ,DISPAT
      RELATIVE  Q8PKUP,Q8PREP
C
      LU = $941
      ITBL(1) = IWORDS
      ITBL(2) = IPG
      ITBL(3) = ILOC
      ASSIGN 500 TO ICOMP
      GO TO (1,2),NDO
    1 CALL WRITE(LU,IZ,ITBL,ICOMP,$44,ITEMP)
      GO TO 3
    2 CALL READ(LU,IZ,ITBL,ICOMP,$44,ITEMP)
    3 CALL DISPAT
  500 CONTINUE
      RETURN
      END
```

```
        NAM         TRAPZ
*
*       UNNEEDED ENTRY PTS PUT HERE
*
        ENT         FOUT
         ENT   RFRMOT,AFRMOT,FLOTIN,RFRMIN,AFRMIN,ASCHX,BINARY
         ENT   DCHX
*
        EQU         FOUT(*)
         EQU   RFRMOT(*),AFRMOT(*),FLOTIN(*),RFRMIN(*),AFRMIN(*),ASCHX(*)
         EQU   BINARY(*)
         EQU   DCHX(*)
*
TRAP    NUM         0
A       JMP*        B
B       JMP*        A
        END
```

C.5. MANIN SOURCE PROGRAM LISTING

```
      PROGRAM MANIN
C
C     ACCEPTS DATA SUPPLIED AT THUMB WHEELS,EDITS,STORES IN TABLE ON
C     DRUM(PAGE 7,WORDS 1501,2000)
C
C     ILTW      POSITION IN LIST (ENTERED ON LEFT THUMB WHEELS)
C     IRTW      DATA VALUE (ENTERED ON RIGHT THUMB WHEELS)
C
C     ICEL MUST LOCATED AT $3F00
      COMMON ICEL(255)
C
      DIMENSION IBUF(20),ITBL(3),ITEMP(8),LIST(2)
C
      RELATIVE LINK,DRUMA,REMARK,ENCODE,FWRITE,DISPAT,RELESE
C
  419 FORMAT (14H ACCEPTED ITEM I4,4X,8H VALUE = I6)
  429 FORMAT (14H          ITEM I4,4X,8H VALUE = I6)
C
      IADR = LINK(0) + ICEL(1)
      IRTW = ICEL(IADR+4)
      ILTW = ICEL(IADR+13)
C
  100 CONTINUE
      IF (ILTW.GT.500) GO TO 410
      IF (ILTW.LT.1)   GO TO 410
      IF (ILTW.EQ.1)   GO TO 215
C
C     READ MDAT(1) FROM DRUM
C
  200 CONTINUE
      CALL DRUMA(1,MM,2)
      IF (MM.EQ.0) GO TO 220
      IF (MM.EQ.1) GO TO 240
      GO TO 410
C
  215 CONTINUE
      IF (IRTW.EQ.0) GO TO 260
      IF (IRTW.EQ.1) GO TO 260
      GO TO 410
C
  220 CALL DRUMA(ILTW,IRTW,1)
      GO TO 420
  240 CALL DRUMA(ILTW,IRTW,2)
      GO TO 430
  260 CALL DRUMA(1,IRTW,1)
      IF (IRTW.EQ.0) GO TO 450
      IF (IRTW.EQ.1) GO TO 460
C
  410 CALL REMARK(1)
      GO TO 600
  450 CALL REMARK(2)
      GO TO 600
  460 CALL REMARK(3)
```

```
      GO TO 600
C
  420 ASSIGN 419 TO IFORM
      GO TO 440
  430 ASSIGN 429 TO IFORM
  440 CONTINUE
      LU = $134F
      ASSIGN 500 TO ICOMP
      LIST(1) = ILTW
      LIST(2) = IRTW
      CALL ENCODE(IBUF,IFORM,2,LIST)
      CALL FWRITE(LU,IBUF,18,ICOMP,$44,ITEMP)
      CALL DISPAT
  500 CONTINUE
C
  600 CALL RELESE(MANIN)
      GO TO 600
      END
```

```
      SUBROUTINE REMARK(N)
C
C     PRINTS A FEW REMARKS
C
C     N          FORMAT SEQUENCE NUMBER
C
      DIMENSION ITBL(3),ITEMP(8),IBUF(23)
C
      RELATIVE  DISPAT,ENCODE,FWRITE
      RELATIVE Q8PKUP,Q8PREP
C
      LU = $194F
C
  100 FORMAT(24H CHECK INPUT - TRY AGAIN /)
  200 FORMAT(18H ACCEPT DATA MODE  /)
  300 FORMAT(16H EDIT ONLY MODE  /)
C
      GO TO (11,12,13),N
   11 ASSIGN 100 TO IFORM
      JJ = 13
      GO TO 20
   12 ASSIGN 200 TO IFORM
      JJ = 10
      GO TO 20
   13 ASSIGN 300 TO IFORM
      JJ = 9
   20 CONTINUE
      ASSIGN 500 TO ICOMP
      CALL ENCODE(IBUF,IFORM,0,0)
      CALL FWRITE(LU,IBUF,JJ,ICOMP,$44,ITEMP)
      CALL DISPAT
  500 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DRUMA(N,MM,NIO)
C
C     READ/WRITE DRUM(PAGE 7,WORDS 1501,2000) DATA TABLE
C                                             ONE WORD AT A TIME
C
C     N          POSITION IN LIST
C     MM         DATA VALUE
C     NIO        1 -WRITE DRUM,  2 -READ DRUM
C
C
      RELATIVE  WRITE,READ,DISPAT
      RELATIVE  Q8PKUP,Q8PREP
C
      DIMENSION ITBL(3),ITEMP(8)
      LU = $941
      ITBL(1) = 1
      ITBL(2) = 7
      ITBL(3) = N + 1500
C
      ASSIGN 500 TO ICOMP
C
      GO TO (1,2),NIO
C
    1 CALL WRITE(LU,MM,ITBL,ICOMP,$44,ITEMP)
      GO TO 3
    2 CALL READ (LU,MM,ITBL,ICOMP,$44,ITEMP)
    3 CALL DISPAT
  500 CONTINUE
      RETURN
      END
```

```
        NAM         TRAPX
*
*       UNNEEDED ENTRY PTS PUT HERE
*
        ENT         Q8QFI,Q8QFX,Q8QFL,FOUT,FLOAT
         ENT   RFRMOT,AFRMOT,FLOTIN,RFRMIN,AFRMIN,ASCHX,BINARY
         ENT   DCHX
         ENT   EOUT
*
        EQU         Q8QFI(*),Q8QFX(*),Q8QFL(*),FOUT(*),FLOAT(*)
         EQU   RFRMOT(*),AFRMOT(*),FLOTIN(*),RFRMIN(*),AFRMIN(*),ASCHX(*)
         EQU   BINARY(*)
         EQU   DCHX(*)
         EQU   EOUT(*)
*
TRAP    NUM         0
A       JMP*        B
B       JMP*        A
        END
```

```
        NAM       CODUM
*
*       RESERVES BLOCK OF COMMON STORAGE - ELIMINATES HAND PATCHING $3FDD
*       CORRELATES DUMMY PROGRAM NAMES WITH LOCATION IN SYSTEM DIRECTORY
*
        COM       DUM(255)
*
*
        ENT       D1,D2,D3,D4,D5,D6,D7,D8,D9
        ENT       D10,D11,D12,D13,D14,D15,D16
*
        EQU       D1($7E),D2($85),D3($8C),D4($93),D5($9A),D6($A1)
        EQU       D7($A8),D8($AF),D9($B6),D10($BD),D11($C4),D12($CB)
        EQU       D13($D2),D14($D9),D15($E0),D16($E7)
*
        END
```