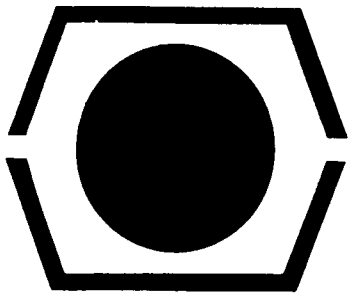


DECUS
PROGRAM LIBRARY

DECUS NO.	12-2
TITLE	PDP-12 UTILITY AND DATA REDUCTION PROGRAMS
AUTHOR	Donald Overton
COMPANY	Eastern Pennsylvania Psychiatric Institute Philadelphia, Pennsylvania
DATE	Submitted: March 2, 1970
SOURCE LANGUAGE	LAP6-DIAL

Abstract

PDP-12 Utility and Data Reduction Programs

by Donald Overton

This tape contains a variety of programs written for the classic LINC or LINC-8 and here modified to run in the PDP-12. Included are data reduction programs which perform auto-correlation, fourier analysis, power spectral analysis and convolution. Utility programs allow selected blocks of LINCtape to be searched, compared or typed out. Also included are programs which allow the user to convert LAP4 or LAP6 manuscripts into LAP6-DIAL, or to disassemble binary code into LAP6 or LAP6-DIAL source. None of these programs were written by the current author, who has simply modified them for operation under LAP6-DIAL in the PDP-12. The tape contains binary and/or source for the programs listed below. Also listed are the original authors. Documents, but no listings, are available.

FREQAN	SIMONS
AUTOCOR	HANCE
QAFILTER	GLAESER
FFOURIER	BRYAN
FFSAMPLE	BRYAN
DATAM	HANCE
FRQANA	ENGEBRETSON
TAPEDUMP	LANAHAN
SEARCH	STEIN
COMPAR	DAVISSON
COMPARE	NICHOLS
LAP4-6	BJERKE
BINLAP6	BJERKE
L8SIMØ	LANBEIN
LAP4Q+A	McDONALD
DSCTABLE	OVERTON
PARAMS	NICHOLS
DIVSUB	STEIN
SPFLT	DILL
QIKDIV	HANCE
RANDOM	LEWIS
NU INDEX	CLAYTON
QANDAS	

PDP-12 Utility and Data Reduction Programs

Introduction

This LINCtape contains a variety of data reduction and utility programs which operate in the PDP-12A. All binaries may be loaded with LAP6-DIAL and are either self-starting, or are started by pressing I/O PRESET and START 20. Documents describing the operation of each of these programs are attached.

All credit for these programs belongs to the original authors. The present author has only modified these programs to run under LAP6-DIAL as a convenience to PDP-12 users.

The question of responsibility for these programs is a difficult one. The original authors obviously cannot be held responsible for correct operation of the modified programs in a new machine. LAP6-DIAL source has been obtained for all data reduction programs to allow program verification and modification.

Any problems encountered with these programs may be referred to:

Dr. Donald Overton
Eastern Pennsylvania Psychiatric Institute
3300 Henry Avenue
Philadelphia, Pennsylvania 19129

Phone: (215) 848-6000, ext. 373

	SOURCE		BINARY	
NAME	BN	BLKS	BN	BLKS
PIP	16Ø	1	251	17
REMAKE		1	47Ø	17
MAGSPY	227		241	1Ø
CONVERT			5Ø7	5
FFSAMPLE	6ØØ	5	55	2
FFOURIER	1Ø3	25	57	7
DIALV2	772	5		
TAPEDUMP	72Ø	52	23Ø	11
COMPARE	3	43	521	7
COMPAR	662	14	514	5
SEARCH			53Ø	6
MARK12			22Ø	7
LAP4-6			211	7
BINLAP6			176	13
FREQAN			536	3
QANDA	541	17		
L8SIM	56Ø	14	173	3
QAFILTER	6Ø5	25	164	7
AUTOCOR	632	17	161	3
DATAM	651	11	574	4
FRQANA	13Ø	21	151	7
LAP4Q+A	77	4		
SPFLT	72	5		
DSCTABLE	66	4		
QIKDIV	46	2		
DIVSUB	53	2		
PARAMS	5Ø	2		
NU INDEX	676	3		
RANDOM	7Ø1	5		
QANDAS	7Ø6	6		
L8SIMØ	714	4		

Title: FREQAN - Frequency Analysis PDP-12

Author: Bill Simons

ABSTRACT

This program will analyze one period of a periodic waveform into 16 Fourier components.

USAGE

Load the FREQAN binary with LAP6-DIAL. Press I/O PRESET and START 20.

If all sense switches =0 a waveform determined by knob 7 is generated (into locations 300-467). Setting SS#0=1 will freeze the waveform, display the Fourier components plotted as a bar histogram, and display the relative value of the component on which the cursor rests. The Fourier components are displayed as a power spectrum: the Y-axis being $(A^2 + B^2)$ where A's are sine and B's are cosine coefficients.

The cursor may be moved (when SS#0 =1) with knob Ø. The position ot the writing may be altered with knobs 4 and 6. In the writing, N= the harmonic that the cursor rests on; XXXX= the relative value of the N^{th} harmonic; and SF = scale factor of the Fourier components. The scale factor is determined by the right three bits of the right switches when SS#Ø =Ø.

RS = Ø is largest scale factor.

RS = 7 is the smallest scale factor.

SUMMARY

SS#Ø = 0 - generate waveform by sampling knob 7.
=1 - freeze waveform and display Fourier components.

Knob Ø - moves cursor over Fourier components when SS#Ø =1.

Knob 4 - Horizontal position of writing when SS#Ø =1.

Knob 6 - Vertical position of writing when SS#Ø =1.

Knob 7 - Determines waveform generated when SS#Ø =1/

Right Switches, least significant digit bits, determine scale factor of Fourier components when SS#Ø =0.

NOTE: The frequency of sampling is fixed and is quite slow.

Title: AUTOCOR Autocorrelation Program PDP-12 with KW12 Clock

Author: James Hance and D. A. Overton*

ABSTRACT

This program samples Channel 10 at 200 Hz for 2.5 sec., displays the data obtained, computes the autocorrelogram for 127 lags (0.625 sec), displays the resulting autocorrelogram and writes it on unit 1 if requested. It may also be used to read and display autocorrelograms previously written on unit 1. Program control is via sense switches.

USAGE

Load AUTOCOR from either unit using the LAP6-DIAL command LOad binary. Press I/O PRESET and START 20. Program operation is controlled by sense switches, as shown in the flow diagram and described below.

METHOD

The program has about 10 functions. Sense switches control function selection and sequence. Each function will be described separately.

Oscilloscope Simulation

Channel 10 is sampled and displayed. No data is stored. The sampling rate provides a 1-sec. sweep rate and is set by the constant DISSET in the program. The program starts in this mode and returns to it whenever SS#0≠0.

Sample Channel 10

When SS#0 =1, the program proceeds from the oscilloscope mode to this sample mode. Channel 10 is sampled 512 times and the data is stored in Q6 and Q7. The constant SAMSET determines the sampling rate. Presently the program samples once every 5 msec. (200 cps.).

Display Data

When data sampling is complete, the data in Q6 and Q7 is displayed for inspection. To reject the data, set SS#0 =0 and the program returns to the oscilloscope mode. To scale the data and compute the autocorrelogram, set SS#1 =1.

Scale Data

The data first undergoes normalization by subtraction of the mean value from all the data points. Following this, they are scaled down to values lying between $\pm 55_8$ to permit the use of single precision multiplication without overflow.

---

*The autocorrelogram computation program is taken directly from COAUTO by James Hance. However data handling has been sufficiently changed so that the program is presented here with a new name and a new document.

Display Scaled Data

The scaled data are briefly displayed for inspection. Then the program proceeds to compute the autocorrelogram.

Compute Autocorrelogram

The correlogram is computed by summing the products of each data point (x_i) and the corresponding point lagging "t" samples in the data array (x(i+t)), where "t" assumes values from 0 to 127. The sum of products is divided by the number of products to give the correlation for that particular lag "t". This process is continued until 128 lag mean products have been assembled in Q5. These are then put into a standard form by being scaled so that the maximum correlation value (at lag zero) is as large as possible but less than +377. This permits convenient display on the scope at a later time. Intermediate points are now interpolated to give effectively 255 lags in the correlogram. A major disadvantage of this program, although not too serious, is the fact that the data array is fixed at 512 points.

COMPUTATION

$$C_{(t)} = \sum_{i=1}^{N-t} \left(x_i \cdot x_{(i+t)} \right) \div \left(N-t \right)$$

where:
$0 < t < 127$ is the lag
$N = 512$
$C_{(t)}$ is the correlation at lag t
x_i, etc. are normalized data values

Display Autocorrelogram

After the computation is complete, the auto correlogram is displayed. It may be either rejected by setting SS#0 =0 or written on tape unit 1 by setting SS#2 =1.

Write Autocorrelogram

Unless otherwise indicated the first autocorrelogram will be written onto block 0 of unit 1, the second onto block 1, third onto block 2, etc. The block number is inserted in address 256 of the correlogram prior to writing, and serves to identify the correlogram during subsequent processing. Hence the structure of a stored autocorrelogram is:

Words 0-255: autocorrelogram
Word 256: Block number I.D.

The BN is stored in loc 17 and is incremented after each block is written. The teletype bell rings when writing is complete.

After block 777 is written, the teletype will type "TAPE FULL" and the computer will halt. Mount a new tape on unit 1 and press START 20 to continue. Writing will commence at block 0 of the new tape.

Normally after autocorrelogram display set SS#2 =1. The correlogram will be written and the block number will be incremented. If you desire to reset the block number, set the desired block number in the right switches and set SS#4 =1. Now when SS#2 =1, the RSW will be read, loc 17 will be set to the indicated block number, the correlogram will be written on that block and the BN will then be incremented in preparation for the next result. Reset SS#4 =0.

Display BN in AC

After the autocorrelogram is written on unit 1, the machine loops with the block number in AC. It is necessary to set SS#0 =0 to get out of this loop and return to the oscilloscope mode.

In review, note that if SS#1 =1 and SS#2 =1, then when SS#0 is set to 1, the program samples, computes, and writes the result on unit 1 without further operator action. Cycle SS#0 to repeat the process.

Read and Display

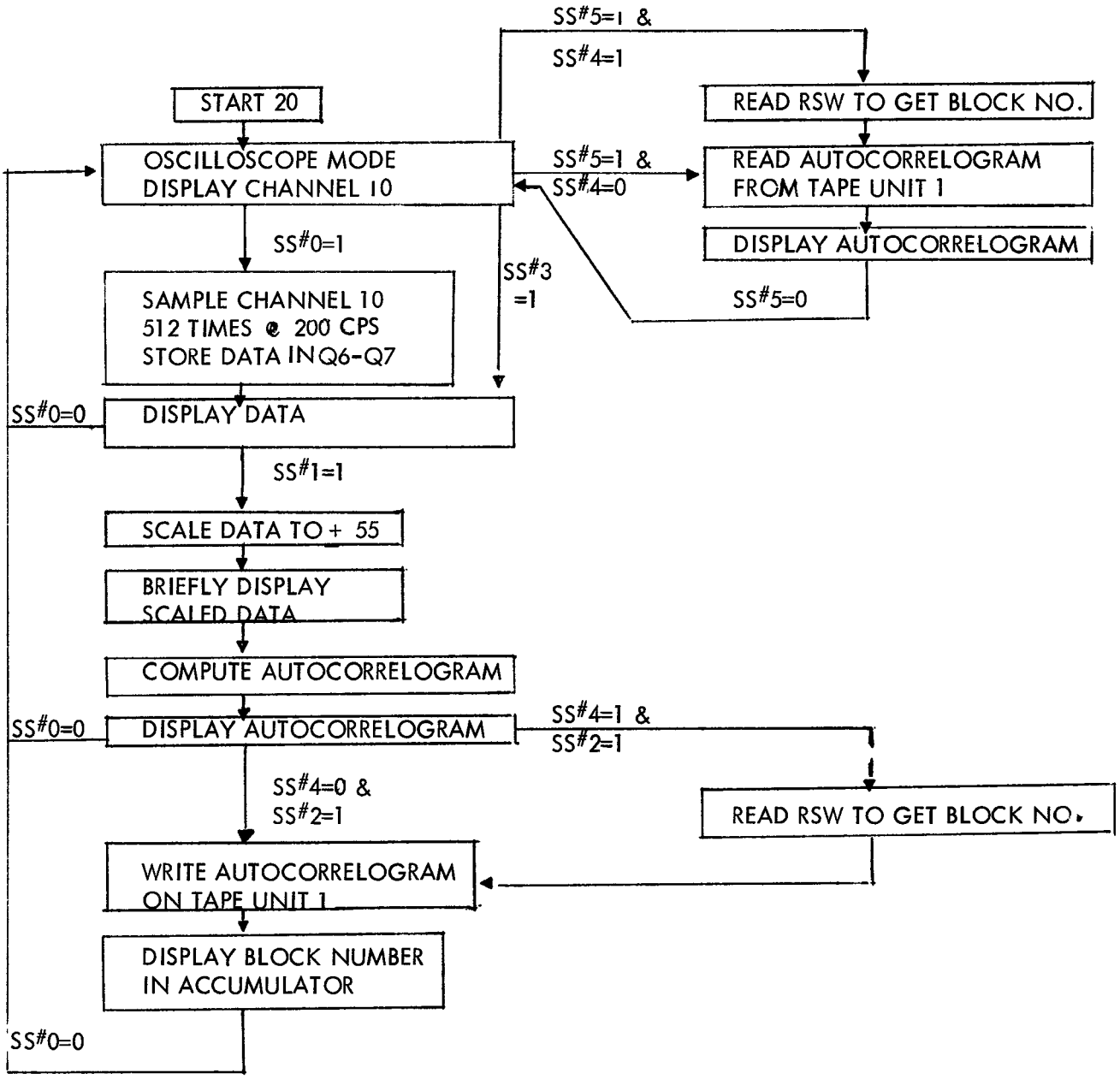
By setting SS#5 =1 while the program is in the oscilloscope mode, the program is made to read the block whose number is in loc 17 and display it. Cycle SS#5 to display the next block. If SS#4 =1 before SS#5 =1, then the BN in the RSW is read and displayed. This may be used to inspect autocorrelograms previously written on unit 1. Set SS#5 =0 to return to the oscilloscope mode.

Omit Data Sampling

If SS#3 =1 while the program is in the oscilloscope mode, any data previously placed in Q6 and Q7 will be displayed. Normally this will be the normalized data from the previous computation. However, other data may be read into Q6-7 manually if desired. If SS#1 =1 after SS#3 =1, the autocorrelogram of whatever you have put in Q6-7 will be computed.

AUTOCOR uses two blocks of data at a time (512_{10} points) and computes an autocorrelogram of 127_{10} lags, i.e., the maximum lag (delta tau) is 25% of the data sample length. It is assumed that the two blocks contain continuous data in time. If they do not, the autocorrelogram will be meaningless.

AUTOCOR FLOW DIAGRAM



Title: QAFILTER PDP-12

Author: Don Glaeser

ABSTRACT

This program performs convolution by multiplication of an impulse response function (IRF) and data. A minor alteration in the program causes it to perform auto- and cross-correlation. For this application the impulse response function and data represent the data to be correlated.

USAGE

Keyboard Control

At the beginning of the program, several questions are presented to the user which are to be answered from the keyboard. Single entry questions should be answered either Y for yes if that option is desired, or N for no. Multiple entry questions must be answered with octal numbers. No protection against non octal characters is provided. For numbers less than 100, leading zeros must be used. The impulse response functions must be located on unit 0 and the data on unit 1. If results are to be stored on tape, they are stored on unit 1 at block numbers requested by the user from the keyboard. No protection against illegal characters is provided. RUBOUT will remove incorrect entries providing they are still on the scope face.

The following questions are asked and should be answered as described:

1) First impulse response function (IRF) block number (on unit 0).

a) For convolution this is self-explanatory

b) For correlation this is the data

The IRF occupies one block of tape and will be read into Q7. Use leading zeros when typing block number.

2) Multiply Option

If only a single IRF is to be used, answer N. If Y is chosen a second IRF may be entered and a display will appear asking its location on unit 0. The product of these two IRF's will be used by QAFILTER as the final IRF or final data for correlation.

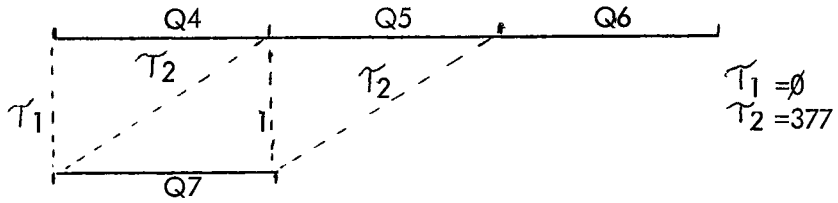
The advantage of this option is that a series of similar IRF's may be easily used in succession, e.g., a damped sine wave when the damping factor varies being represented by a series of exponentials.

3) Prediction

The complete convolution function or correlation function is too large to be computed and displayed at once. Therefore, this option determines what portion of the final function will be computed. Answer Y or N. An answer of Y will produce a question asking what value of τ is to be used. Do not confuse this τ with the delay value of the correlation function normally

called τ. For example, if the user is interested in the correlation function at delay = 0 then prediction should be zero.

In this program data occupies 3 blocks (Q4-Q6). (For correlation it is best if each block is one period.) The method of convolution (correlation) begins multiplication according to the τ value of prediction. When $\tau = x$, multiplication begins by multiplying the xth point of the IRF by the x+1th point of the data.



4) Inversion

a) The method used for convolution requires that the IRF function be inverted. You may store inverted IRF functions on tape in which case answer N to the inversion option. If the IRF's are not inverted, answer Y.

b) Data should not be inverted if cross- or autocorrelation is being performed.

5) Computation Speed

Choosing fast computation (Y) doubles the speed by skipping every other point. Answer N for slow computation.

6) Data

Data occupies three blocks. It is best to have 3 blocks of consecutive data. If this is not available, fill all 3 lines with the same block number. For correlation all three blocks may be the same. For auto-correlation, of course, data will be the same as the IRF. Data is destroyed during computation and is read in again before recomputation. Leading zeros must be used. Push RETURN after each data BN is typed.

After the third BN is filled in the computation begins. The computation depends critically on the scale factor set by knob 6. If the SF is too high or too low, the scope may display a straight line as the result. You can get some idea of whether the result will be too high or too low by watching the moving dot during computation. SS#5 is active during the computation. If the displayed result is too large or too small, set SS#5 =1, adjust the scale factor with knob 6, and set SS#5 =0 to try again. When the computation is complete the display of data, IRF, and result will appear.

IN THE DISPLAY MODE:

Knob 1: Varies trace separation. Commonly from top to bottom arrange: data, IRF, output.

Knob 2: Display scale factor of IRF.
SS#2: Temporarily kills display of time inverted IRF.

Knob 0: Display scale factor of output (result).
SS#1: Temporarily kills display of output.

Knob 4: Display scale factor of data. 2nd block of data if no prediction
SS#3: Temporarily kills display of data.

When in the data-display mode, typing the letter O produces an options display.

```
   DO ?
0. RETURN TO DISPLAY
1. REPEAT
2. NEW FILTER
3. NEW DATA
4. INDEX DATA
5. COMP SPEED OPTION
6. STORE RESULT
```

Select an option and press RETURN

Option 0: Returns to an unchanged display of data, IRF, and results.

Option 1: Repeats the computation. If SS#5 =1, a new scale factor (SF) may be selected by adjusting knob 6. When SS#5 is set to zero, recomputation will occur.

Option 2: Restarts the entire program. All the initial questions must be reanswered.

Option 3: Allows you to select new data. The filter remains unchanged.

Option 4: Index Data. If the 3 data blocks were sequential, each data BN is increased by 1.

Option 5: Allows change in computation speed. Data BN's must be reentered.

Option 6: Allows you to specify the BN on unit 1 where you wish the result written.

NOTE: In order to perform correlation with QAFILTER two changes must be made in the source which must then be reassembled.

Alternately the binary may be changed directly.

Source Line	Octal Location	Contents for Convolution	Contents for Correlation
1326	5701	ADD F7 (2035)	NOP (0016)
76	4074	F1, 6001	F1, 2000

Title: LINC SPECTRUM PROGRAM (Using New Cooley-Tukey Fast Fourer Transform Algorithm) PDP-12 with KW12 clock

Author: James Bryan
National Institute of Mental Health

ABSTRACT

The version of the program used at the National Institutes of Health utilizes a standard 2000 word LINC III with a Datamec tape deck and a digital plotter. Use of the Datamec permits processing continuous epochs of any length. Since a program to work with this particular input-output equipment would be of use to no one else, a modified version was prepared to share with PDP-12 users.

The modified version uses LINCtape for storage and the oscilloscope display for output. Since the LINC III has unbuffered tape, the longest epoch which this version can handle is 7 quarters long or 1792 samples. PDP-12 users can easily modify the digitizing program for epochs of any length since their tape system is buffered. The LINCtape distributed has been modified for use on the PDP-12. The only restriction is that the number of data blocks in an epoch must be odd.

FFSAMPLE is a data sampling program specifically designed for use with the program FFOURIER. It samples channel 10 for 17.5 sec at 102.4 samples per sec., stores the data in Q1 — Q7, and then displays the data. When SS#0 is set to 1, the data is written onto unit 1 at blocks 0, 5, 6, 13, 14, 21, and 22, and the program FFOURIER is read into core from BN60 on unit 0. Certain constants are transferred from FFSAMPLE to FFOURIER and the FFOURIER program proceeds to completion and displays the resulting autocorrelation.

USAGE

FFSAMPLE may be loaded with LAP6-DIAL. Alternately set LS=0700. RS-OXXX and press I/O PRESET and DO. XXX is a block number 1 greater than the starting BN shown for the FFSAMPLE binary in the LAP6-DIAL index. Data sampling begins immediately after START 20. When sampling is complete and the data is displayed, set SS #0 =1 to compute the spectrum.

FFOURIER is the spectrum computation program. It is identical to DECUS L-25. The program must reside at BN60-65 on unit 0 in order to be properly loaded by FFSAMPLE, and it has been so placed on this LAP6-DIAL tape (BN57 is a header block-not part of the FFOURIER binary which begins at BN60). During operation FFOURIER writes on allunit 1 tape blocks 0-22 which were not already written on by FFSAMPLE. Hence a scratch tape should be placed on unit 1. When the computation is complete the spectrum will be displayed and may be scaled as follows:

To scale display up by 2, hit U key; this may be done repeatedly.
To scale display down by 2, hit D key; this, too, may be repeated.
To reset scale, hit S key.
Sense switch 0 produces a bar-chart display.

Notes: The location of the program on unit 0 may be changed. Also, the region of unit 1 for storage may be changed. The following notes may help to implement such changes:

FFSAMPLE writes on tape unit 1 at the following blocks:

$$N,\ N+5,\ N+6,\ N+11_{(10)},\ N+17_{(10)},\ N+18_{(10)}$$

where N is the initial block number. N is a program parameter located in memory address 22 of FFSAMPLE and can be changed to any arbitrary block. The unconventional spacing of the data blocks provides working space for the spectrum program with minimum tape shuffling.

This particular digitizing timing is intended for EEG work from 0.4 to 40 Hz. It should be used with a low-pass filter which is flat from zero to 40 Hz and at least 40 db down at all frequencies beyond 60 Hz.

Input data must be scaled in range ± 1777 before storage on unit 1.

The Spectrum Program proper resides on blocks 60-65 on unit 0. Blocks 60-63 contain the program, and blocks 64 and 65 contain a sine table. To compile the program for other locations, equalities 4A and 4C must be changed accordingly. Memory locations 22 and 23 are parameters; 22 determines the initial data block number on unit 1, and 23 determines the number of blocks to be processed; in general . . .

$$(23) = \frac{n-1}{2}$$

where n is the number of blocks and must always be odd.

Calibration marks on the display are at 5 Hz intervals. The points at 0.0 Hz and 0.2 Hz have been suppressed since the d-c term usually makes these overflow first. Points are at 0.2 Hz intervals.

Binary location 52 (block 60) contains a scaling code. Normally it should be 377. It may be changed to 17.7 or 77 at some risk of overflow.

Title: DATAM

Author: James Hance

ABSTRACT

This program retrieves, displays, and stores data and provides filtration, intergration, differentiation, and size variation features.

USAGE

Load DATAM with LAP6-DIAL. Press I/O PRESET and START 20.
Mount the data tape on unit 1.

To retrieve data: Type R___ where___ is the block number containing a waveform.

At anytime while data is being displayed, any of the following operations may be performed:

Typing:

F will smooth the waveform; low-pass filter (moving average)

H is a high-pass filter

D will differentiate the waveform

I will integrate the waveform

L will increase the absolute amplitude of the waveform

S will decrease the absolute amplitude of the waveform

A will add the contents of the right switches to each point of the waveform effectively raising the DC level

P will invert the saveform (i.e., change polarity)

B will plot the waveforms as a bargraph - no other functions may be pertormed when in the bargraph mode of plotting

C will return to normal operating mode

O will restore the original waveform

Points of caution:

1) Operations which overflow the scope's edges or which reduce the waveform to zero may effect an error. Care should be taken to avoid these problems.

If either of these events does occur, typing O will restore the original waveform.

2) After typing R or K; three octal numbers must be typed. DATAM does not check this and will read or write an incorrect block if leading zeros are not present when needed, or decimal numbers or illegal characters are struck following R or K. Data is read from unit 1 and results are written on unit 1.

Title: FRQANA Frequency Analysis PDP-12

Author: A. M. Engebretson

ABSTRACT

This program performs frequency analysis resulting in sixty-four component cosine, sine, and rms spectra. Spectrum displays are scalable. Resynthesis from the spectra can also be performed.

USAGE

1. Load FRQANA with LAP6-DIAL. Press I/O PRESET and START 20.

2. Read in data for frequency analysis.

Type RXXX where XXX is the block number on unit 1 containing the data. Leading zeros must be used.

Before typing R, the following sense switches are operative and affect the read in of data.

SS #4 = 0 and SS#5 =1: read in XXX once
SS #4 = 1 and SS#5 =0: read in XXXtwice
SS #4 =0 and SS#5 =0: read in XXX and XXX+1
SS #4 =1 and SS#5 =1: DO NOT USE

After the data has been read in, it is displayed. The data display is 512 points. The frequency analysis occurs over 512 points.

3. The data display may be scaled by typing S $\pm$ N where N is a number between 1 and 7. The scale factor is:

$$\frac{2N \pm 1}{2^N}$$

Note that the largest change in amplitude is obtained with S$\pm$ 1 and that scaling may be repeated if necessary.

4. Knob 1 is a trace separator, affecting only the vertical display position of the data. When fully counterclockwise, the top trace is the original data. The second trace will be the resynthesized waveform after the frequency analysis and resynthesis have been performed.

5. To perform the frequency analysis, type F. When the analysis is complete, the display corresponding to SS #Ø and #2 settings will appear.

6. The frequency component display may be obtained by putting SS #Ø =1. The display is:

Cosine Components Sine Components

These are scalable by the right six bits of the right switches.

8. To compute the resynthesized waveform from the frequency components, type T. Again, the display will appear according to SS #Ø and #2 settings. Note that SS #0 and SS #2 affect only the display, not the computations.

SUMMARY OF OPERATIONS

Knob 1 - Trace separator. When knob 1 is fully counterclockwise the top trace is the input data, the bottom trace is the synthesized data.

SSW 0 =0 - time function display.
=1 - spectrum display. SSW2 operative.

SSW 2 =0 - cosine and sine spectra are displayed.
(cosine on the left, sine on the right).
=1 - rms spectrum displayed.

SSW 4&5 - Control data entry, as described above.

Right switches - right six bits scale cosine-sine spectra display
left six bits scale rms spectrum display.

MEMORY ALLOCATION

0----576	Program
600----677	Computed sine spectrum
700----777	Computed cosine spectrum
1000---1777	Synthesized time function
2000---2777	Input data
3000---3777	Cosine table

LAP6-DIAL manuscript structure

Lines 0-640	Computational Program
640-1650	Cosine table for 3000-3777
1650-2240	L8SIM for segment 0

Title: TAPEDUMP PDP-12A

Author: Marilyn Lanahan
Laboratory Computer Facility, University of Wisconsin
Supported by N.I.H. Research Grant FR00249 (SRRB)

PURPOSE

TAPEDUMP will print the contents of LINCtape with octal, unsigned or signed decimal (or octal and incorrect alphanumeric conversions). The output device must be a ASR33 teletype.

OPERATIONAL PROCEDURE

START PROCEDURE

Start LAP6-DIAL

→ LO TAPEDUMP, U↓ U=Unit

I/O PRESET

START 20

The flow of the program is guided by questions displayed on the oscilloscope and answered through the keyboard. For all questions which require a numerical value for an answer, each of the following is true:

a. The response must be an octal number.

b. Leading zeros are not required.

c. A response consisting of all blanks is equal to 0.

The first display asks the location and number of blocks to be dumped:

```
PRINT??? BLOCKS
FIRST BLOCK ???
UNIT ?

QUARTER ?

ID IS ????????
TAPE NO. IS ?????
```

Any number of blocks $000\text{-}777_8$ may be requested starting at any block $000\text{-}777_8$ on units 0 or 1. A request of 000_8 blocks acts as no operation and the program proceeds to the PRINT MORE? display described below. The quarter requested may be 0-7 and will determine in which memory locations the first block appears to be stored. Successive blocks will appear to be in successively

higher locations through 7777_8. The identifier may be any 8 alphanumeric characters and the tape number any 5 alphanumeric characters; each will be printed with every block. Press RETURN after filling each blank. RUBOUT will remove errors. The last RETURN proceeds to the next display.

The second display determines the type of conversion desired:

```
IN?

0 OCTAL
1 UNSIGNED DECIMAL
2 SIGNED DECIMAL
3 ALPHANUMERIC
```

If the alphanumeric conversion is chosen, the output will consist of both octal values and the alphanumeric equivalents. The alphanumeric characters typed are appropriate for the LINC TTY and hence inappropriate for programs assembled under LAP6-DIAL. Hence option 3 can be used to scan LAP6 or LAP4 tapes but is useless for LAP6-DIAL source. Use MAGSPY instead.

Printing will proceed after the above display is terminated. The printing may be terminated at any time by striking EOL. The following display appears when the output has been stopped due to EOL or completion of the requested blocks:

```
PRINT MORE?
```

A response of Y will cause TAPEDUMP to restart at the display which determines the number and location of blocks to be dumped. Any other response causes LAP6-DIAL to be loaded and started.

TELETYPE OUTPUT

The output consists of one or more LINCtape blocks per page with proper headings. Each block begins with the line:

DUMP of AAAAAAAA LINCTAPE NO nnn BLOCK bbb

where AAAAAAAA is ID, nnn is tape number and bbb is the block number in octal.

The printing of the tape contents follows. Each line consists of the octal location of the first word on the line followed by the contents of the next 8 locations. The contents are converted to octal or signed or unsigned decimal nos. as requested. In the case of alphanumeric conversion, the contents of each word is printed in octal on the left of the page with their alphanumeric interpretation on the same line at the right. Character conversions are for LINC TTY.

If the block has never been written in and hence contains a mark pattern, the following message will appear after the heading:

LINCTAPE BLOCK nnn contains a mark pattern

where nnn is the block number in octal. The contents of the block will not be printed.

If the block contains a group of 2 or more lines of IDENTICAL data values, the following message will appear in place of these lines:

LOCS mmm THRU nnn ALL = xxx

where mmm is the octal location of the beginning of the repetitions, nnn the octal location of the end of the repetitions, and xxx the value of the data converted to octal or signed or unsigned decimal as requested. Only whole lines of data are considered in forming the condensed message above. Nonrepetitious data will be printed out normally.

If the block could not be read without a parity (checksum) error, the following message is inserted:

**** LINCTAPE ERROR PERSISTS AFTER 5 REREADS, CHECKSUM = xxxx***

where xxxx is the LINCtape checksum in octal.

Input and Output Mounting

TAPEDUMP may be loaded from any unit (however control returns to LAP6-DIAL on unit 0 when TAPEDUMP terminates). Blocks from any unit may be printed.

Timing

Using the teletype: a maximum of 8 minutes per block.

Equipment Configurations

PDP-12A with ASR33 TTY

METHOD

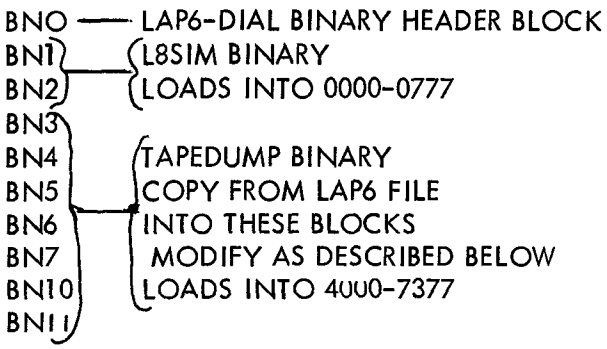
TAPEDUMP reads LINCtape into memory and converts one line of print at a time. Different conversions are selected by inserting JMP instructions to the proper routine.

METHOD FOR PRODUCING A TAPEDUMP LAP6-DIAL BINARY

Make a LAP6-DIAL index entry for TAPEDUMP as follows: Put L8SIM in the working area. Modify L8SIM for direct loading into loc 0000-0777 as described in sect. 6.4 of the L8SIM document. Add to the manuscript as follows:

```
SEGMNT 2
  HLT
 *400
  HLT
 *1000
  HLT
 *1400
  HLT
SEGMNT 3
  HLT
 *400
  HLT
 *1000
  HLT
```

Assemble this source and save binary using the name TAPEDUMP. A 12 block binary index entry should result. These blocks are utilized as follows:



Obtain a LAP6 TAPEDUMP binary by assembling the TAPEDUMP manuscript using LAP6. DECUS No. L-37 contains the TAPEDUMP manuscript and binary. Copy the TAPEDUMP LAP6 binary into relative blocks 3-11 of the space reserved for TAPEDUMP on the LAP6-DIAL tape.

Modify the TAPEDUMP LAP6-DIAL binary directly using DO-RDC and DO-WRC commands. The following changes are required:

BN0, 1, 2 No changes

BN3

Location	Old Contents	New Contents	Mnemonic
0020	0061	1020	LDAI
0021	7377	1200	1200
0022	0062	0004	ESF
0023	2777	6044	JMP44

BN4

251	0075	0076	SET I 16
253	0076	0077	SET I 17
255	6015	6016	JMP 16

BN5 No changes

BN6

0014	0504	0016	NOP
0161	0502	0016	NOP
0276	6020	6044	JMP44

BN7, 10, 11 No changes

Title: SEARCH Masked Word Search PDP-12A

Author: Mark Stein

ABSTRACT

This program searches blocks of tape for a word called the "search word." The search word and a mask are determined by the user. Single or successive tape blocks may be searched.

USAGE

Load the SEARCH binary with LAP6-DIAL. Press I/O PRESET and START 20. The following will be displayed:

```
SEARCH
BLOCK ??? UNIT ?
FOR QUARTER ?
```

Type the number of the block to be searched, RETURN, the unit number (zero or one), and another RETURN. If successive blocks are to be searched, type the number of the first block. Leading zeros must be used.

Then type the memory quarter (0 - 7) for which the block (or initial block) being searched is intended during program operation.

Typing RETURN presents the following display:

```
FOR QUANTITY ????
MASKED BY ????
```

Type in the quantity, i.e., search word, then RETURN, the mask digits, then RETURN.

Only the numbers 0 - 7 are accepted and leading zeros must be used.

The Mask

During the search the "search word" is compared to each word on the block being searched. For every bit of the mask that is a zero, it is assumed that the corresponding bit of the search word has been found. Logically, the mask is "anded" with the difference between the search word and each word of the block being searched.

Examples

Search word	1020	1020	0443	0443	anything
Mask	7777	7757	7777	7770	0000

Finds all	1020	1020 1000	443	440 441 442 443 444 445 446 447	Finds all words on specified block
Purpose	Find all	Find all	Find all	Find all	Look at every word
	LDAi	LDA and LDAi	SNS 3	SNS N	

If an error has been made in typing the parameters, the following error message will be displayed:

```
ERROR EOL
FOR OPTIONS
```

Press RETURN to start over.

If no error has been made, the search commences. When a masked search word has been found, the following display appears:

```
BLOCK    LOC     WORD
 XXX     NNNN    YYYY
```

XXX is the block of tape being searched.

NNNN is the address in memory which contains the "word found." The location is determined by the position of the word within the tape block and the quarter for which the tape block is intended.

YYYY is the "word found."

With SS #5 =0 typing RETURN continues the search of this block.

With SS #5 =1 typing RETURN discontinues the search of this block.

If SS#0 =1 when the search ends, LAP6-DIAL will be restarted.

If SS#0 =0 when the search ends, the following display will appear:

```
OVER AND DONE
```

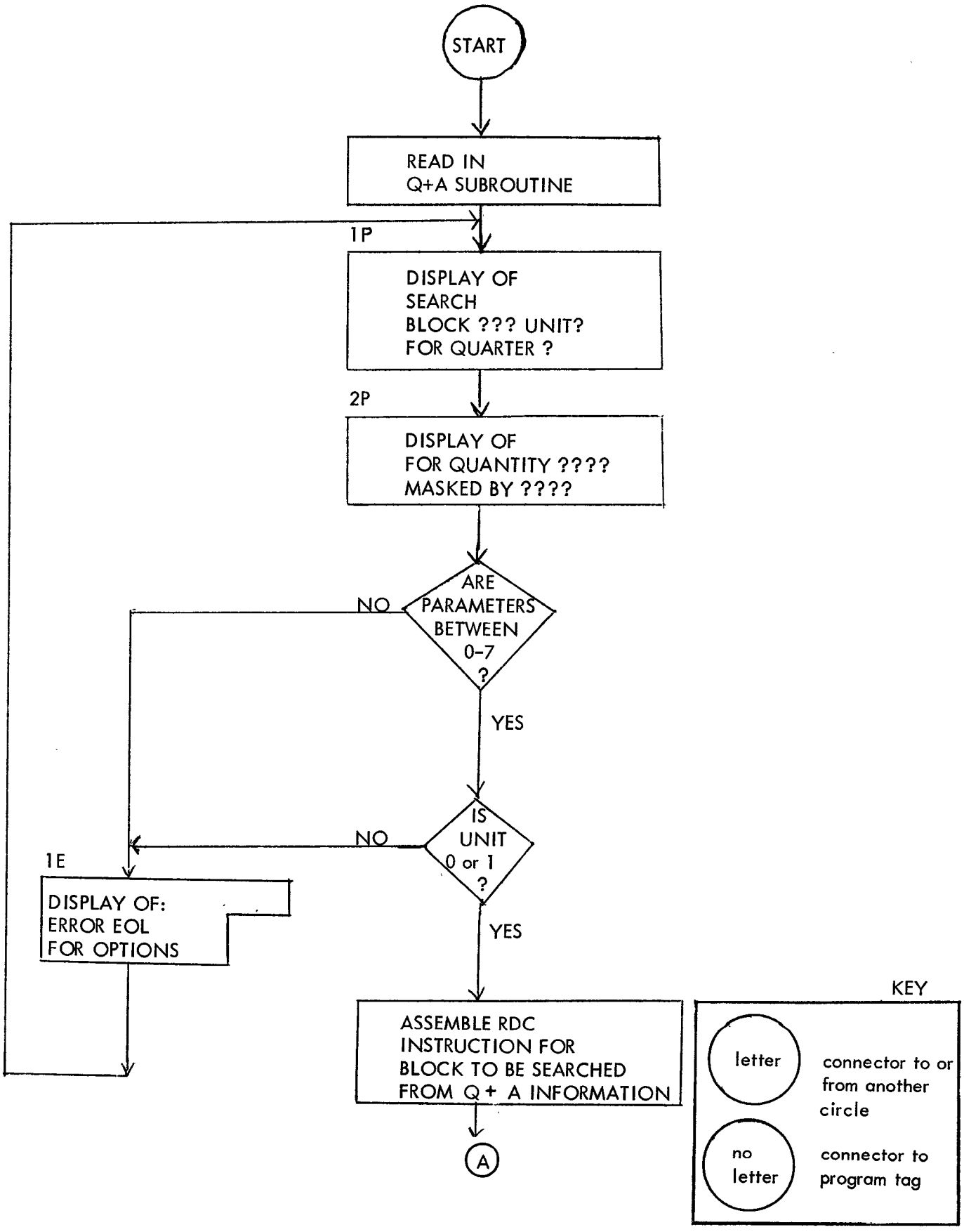
Typing RETURN

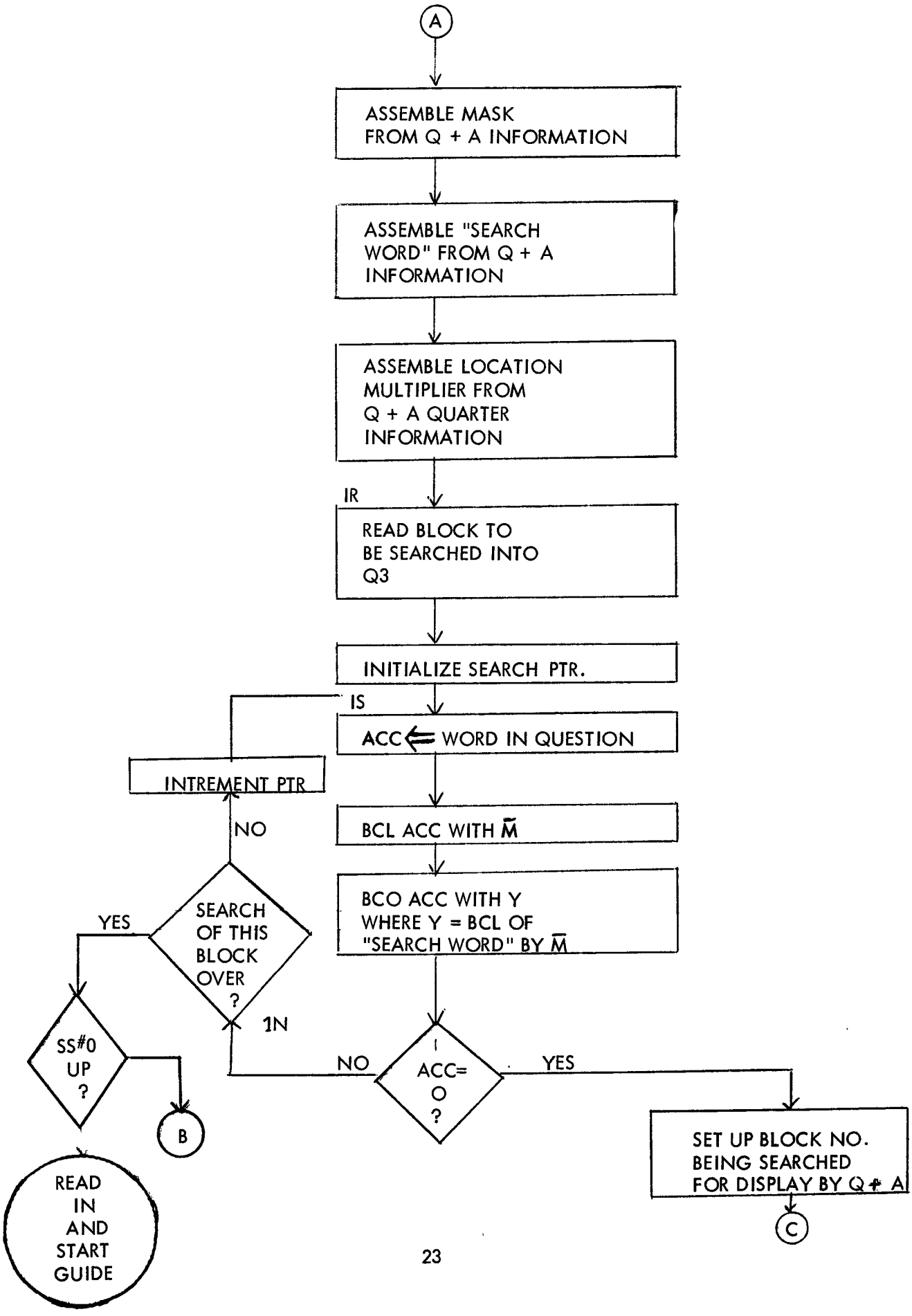
With SS#4 =0 restarts the SEARCH program and option display.

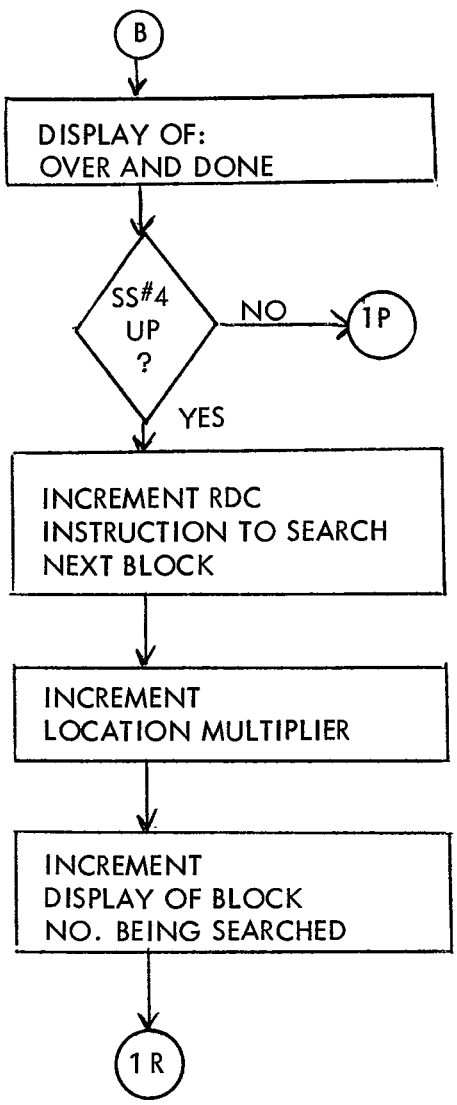
With SS#4 =1 — the block just searched is incremented and the next block is searched. It is assumed that the block to be searched belongs in the next quarter of memory. Therefore, the location number will be appropriately increased when a "find" occurs.

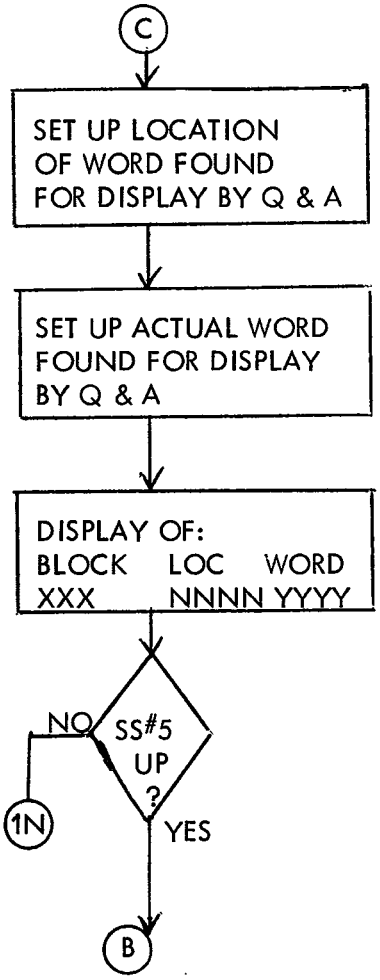
Summary of Operation

SS#0 =0	restarts SEARCH when the search is completed.
=1	restarts LAP6-DIAL when the search is completed.
SS#4 =0	returns to option display.
=1	actuates search of next block.
SS#5 =0	continues search of each word in tape block.
=1	terminates search of the tape block and acts according to SS#0 and SS#4.


START
READ IN
Q+A SUBROUTINE
1P
DISPLAY OF
SEARCH
BLOCK ??? UNIT?
FOR QUARTER ?
2P
DISPLAY OF
FOR QUANTITY ????
MASKED BY ????
ARE
PARAMETERS
BETWEEN
0-7
?
NO
YES
IS
UNIT
0 or 1
?
NO
YES
1E
DISPLAY OF:
ERROR EOL
FOR OPTIONS
ASSEMBLE RDC
INSTRUCTION FOR
BLOCK TO BE SEARCHED
FROM Q + A INFORMATION
A
KEY
letter
connector to or
from another
circle
no
letter
connector to
program tag


A
ASSEMBLE MASK FROM Q + A INFORMATION
ASSEMBLE "SEARCH WORD" FROM Q + A INFORMATION
ASSEMBLE LOCATION MULTIPLIER FROM Q + A QUARTER INFORMATION
IR
READ BLOCK TO BE SEARCHED INTO Q3
INITIALIZE SEARCH PTR.
IS
ACC ⇐ WORD IN QUESTION
BCL ACC WITH M̄
BCO ACC WITH Y WHERE Y = BCL OF "SEARCH WORD" BY M̄
ACC= O ?
NO
YES
SET UP BLOCK NO. BEING SEARCHED FOR DISPLAY BY Q + A
C
INTREMENT PTR
NO
SEARCH OF THIS BLOCK OVER ?
YES
1N
SS#0 UP ?
B
READ IN AND START GUIDE


B
DISPLAY OF:
OVER AND DONE
SS#4
UP
?
NO
1P
YES
INCREMENT RDC
INSTRUCTION TO SEARCH
NEXT BLOCK
INCREMENT
LOCATION MULTIPLIER
INCREMENT
DISPLAY OF BLOCK
NO. BEING SEARCHED
1 R


C
SET UP LOCATION
OF WORD FOUND
FOR DISPLAY BY Q & A
SET UP ACTUAL WORD
FOUND FOR DISPLAY
BY Q & A
DISPLAY OF:
BLOCK LOC WORD
XXX NNNN YYYY
NO
SS#5
UP
?
1N
YES
B

METHOD FOR MAKING A PDP-12 SEARCH BINARY

1. Make a LAP6-DIAL index entry for SEARCH as follows: Put L8SIM in the working area. Modify for direct loading into segment 0 as described in Sect. 6.4 of the L8SIM document. Add to the manuscript as follows:

```
SEGMNT 2
  HLT
*400
  HLT
*1000
  HLT
```

2. Assemble the source and save binary using the name SEARCH. A 6 block binary index entry should result. These blocks are utilized as follows:

BNO -- HEADER - created by LAP6-DIAL

BN1, BN2 — L8SIM binary assemblies for loading into 0000-0777

BN3, BN4 — SEARCH binary from LINC-8 library tape modified for the PDP-12

BN5 -- Q+A binary from BN403 of the LINC-8 library tape

3. Copy the 2 block SEARCH binary from the LINC-8 library tape into relative blocks 3-4.

4. Copy the Q+A binary from BN 403 of the LINC-8 library tape into relative block 5.

5. Modify the resulting binary directly using DO-RDC and DO-WRC instructions.

BN0 – 2 No change

BN3

Location	Old Contents	New Contents	Mnemonic	Function
0012	0	1020	LDA I	/SET TRAP
0013	0	1200	1200	
0014	0	0004	ESF	
0015	0	6022	JMP22	/START SEARCH
0016	0	0701	RDC	/START LAP6-DIAL
0017	0	7300	7300	
0020	0700	6012	JMP 12	/START PGM
0021	2403	0000	HLT	NEVER HALTS

BN4

0411	0700	0006	DJR	/JMP TO
0412	3400	6610	JMP 610	RESTART LAP6-DIAL
0610	0	1020	LDAI	/RESTART
0611	0	0701	0701	/DIAL IF
0612	0	4016	STC16	/SS#0=1
0613	0	1020	LDAI	
0614	0	7300	7300	
0615	0	4017	STC17	
0616	0	6016	JMP 16	

BN5 No changes

Title: COMPAR Compare Contents of Tape Blocks PDP-12

Author: Sharon R. Davisson

ABSTRACT

This program will compare the contents of tape blocks.

USAGE

After loading the program with LAP6-DIAL, press I/O PRESET and START 20. The following will be displayed on the scope:

```
COMPARE
BLOCK ??? ON UNIT ?
TO BLOCK ??? ON UNIT ?
FOR ??? BLOCKS
```

Answer the questions. Leading zeros are not necessary in any of the fields.

The program will begin comparing with the initial specified blocks, automatically increment each block number by one, and continue comparing until the number of blocks requested have been compared.

If the comparison is completed satisfactorily, the following will be displayed.

```
ALL OK
```

Press RETURN, and SSW#0 becomes operative in the following manner:

SSW#0 =0 - the program will return to LAP6-DIAL
SSW#0 =1 - the program will return to the first display in COMPAR.

If the comparison is unsatisfactory, the following will be displayed:

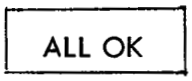
```
BAD COMPARE !

 LOC   BLK   CONTENTS
nnnn   nnn     nnnn
       nnn     nnnn
```

"nnnn" under LOC will be the location relative to Q0 where the difference is found. The blocks and differing contents will be displayed under their proper headings. If tape unit 1 is referenced, a "u" will appear behind the appropriate block number. When the EOL key is struck, SSW#1 becomes operative in the following manner:

SSW#1 = 0 - the rest of the block will be compared.

SSW#1 = 1 - the rest of the block will not be compared. The program will either go to the next set of blocks to be compared or if no more blocks are to be compared,



will appear on the scope and SSW#0 will become operative (see above).

Title: COMPARE PDP-12

ABSTRACT

COMPARE will compare the contents of two sets of LINCtape blocks. Errors are either displayed on the LINC scope or printed on the teletype.

USAGE

Program Implementation

COMPARE may be loaded from any unit via the LAP6-DIAL LOad meta-command. Press I/O PRESET and START 20.

Space Required

The binary requires 7 blocks in the file. COMPARE uses all of lower and upper memory during execution.

Operational Procedure

At all times during the operation of COMPARE the following 3 keys will be recognized:

D Place the program in display mode, i.e., display on the scope any errors encountered. Initially COMPARE is in the display mode and will stay in the display mode until P is struck.

P Place the program in print mode, i.e., print on the teletype any errors encountered.

L Return to LAP6-DIAL on unit 0 immediately.

In addition, the following keys will be recognized during the display or printing of compare errors:

Ø Skip any further errors in this block.

1 Move forward to the next error.

Q Return to the first error in this block.

W Move backwards to the previous error within this block.

S Stop the compare and present display (2) or (3).

Upon entry to COMPARE, the following display appears:

(1)

```
COMPARE ???? BLOCKS
FROM BLOCK ??? UNIT ?
  TO BLOCK ??? UNIT ?
```

All responses must be in octal. Leading zeros are not required. Units 0 and 1 may be used. COMPARE will not allow a set of blocks to wrap around the end of tape, e.g., 2 blocks starting at 777.

The program will begin comparing with the initial specified blocks, automatically incrementing the block numbers by one until the requested number of blocks have been compared. If an error occurs, the program will either display or print the discrepant information (see Displays and Teletype Output).

When all requested blocks have been compared, or when S is struck, one of the two following displays appear. Strike L to return to LAP6 on unit 0 or any other key to return to display (1).

(2)

```
COMPARE DONE
NO ERRORS
```

(3)

```
COMPARE DONE
  ERRORS
```

Displays

If an error is encountered while the program is in display mode, the following display will appear:

```
BAD COMPARE
LOC# BLK CONT
nnn u bbb xxxx
    u bbb yyyy
```

nnn is the location, reterenced to the beginning of the block, where the bad compare is made. The unit number, block number, and differing contents will be displayed under their proper headings. Operational Procedure gives details on the control characters affecting this display.

Teletype Output

If an error is encountered while the program is in print mode, the following typeout will appear. The unit/block number heading appears once per block.

```
          %u    %u
       BNbbb    bbb
LOC
nnn xxxx  yyyy
qqq zzzz  aaaa
             .
             .
             .
```

After each line is printed, the program goes forward to the next error automatically.

Timing

If no errors occur, COMPARE takes approximately 35 seconds to compare two sets of 100_8 blocks each, assuming the two sets are not on the same unit.

Software Configuration

COMPARE returns to a standard configuration; LAP6-DIAL on unit 0.

METHOD

If the two sets of blocks requested come from the same unit, then the blocks are read in an alternating fashion. Take the example of comparing 5 blocks from block 200 on unit 0 to block 500 on unit 0. They would be read in the following order (COMPARE always tries to read 2 consecutive blocks from each set):

```
┌ 200 ┐
└ 201 ┘
┌ 500 ┐
└ 501 ┘

┌ 502 ┐
└ 503 ┘
┌ 202 ┐
└ 203 ┘

┌ 204 ┐
└ 504 ┘
```

This results in a savings on tape motion time. The Q & A used by COMPARE is self-contained and is modified during execution.

CONVERSION OF LAP4 TO LAP6

Title: Conversion of LAP4 to LAP6 PDP-12

Author: Conrad C. Bjerke and Marilyn Lanahan
Biomedical Computing Division,* University of Wisconsin

PURPOSE

LAP4-6 will convert a LAP 4 manuscript into a LAP6 manuscript. LAP6-DIAL users may then use CONVERT to obtain a LAP6-DIAL source program.

OPERATIONAL PROCEDURE

Load the LAP4-6 binary from either unit using the LAP6-DIAL LOad-binary command. The initial questionnaire should appear on the scope.

Mount the LAP4 manuscript tape on unit 1 (write-locked).

Mount a scratch tape on unit 0 (write-enabled).

LAP4-6 writes only on blocks within the LAP6-DIAL working area. Hence you may use a LAP6-DIAL tape as the scratch tape providing the working area has been cleared.

Now fill in the following questionnaire:

```
LAP4 to LAP6
LAP4 MS IN BLOCK ???
ON UNIT 1
```

Type the correct block number on the keyboard; then depress RETURN. RUBOUT may be used to delete incorrect entries.

If there is no LAP4 manuscript beginning at the specified block number on unit 1, the display:

```
NO MS
```

will appear. Press RETURN to return to the first display.

When the translation of the manuscript is finished, the display:

* LAP4-6 was developed by the Laboratory Computer Facility at the University of Wisconsin Medical Center under grant FR00249-04 from the Division of Research Facilities and Resources of the National Institutes of Health.

```
LAP6 MS AT BLOCK 370
ON UNIT 0
```

will appear. The new LAP6 manuscript is now complete and can be used. If you want to start over, press RETURN. If the program is too long, that is, if it extends beyond block 466 on unit 0, the display:

```
MS TOO LONG
```

will appear. Note that the manuscript so converted is incomplete.

Press STOP. If the conversion was complete the LAP6 manuscript now starts at BN 370 on unit 0. If the manuscript was too long, the first 76 blocks of it are found in blocks 370-466 on unit 0.

Place the SCRATCH tape on unit 1 (write-locked).

Place the LAP6-DIAL tape containing the CONVERT binary on unit 0 (write-enabled).

Start LAP6-DIAL and clear the LAP6-DIAL working area.

LOad the CONVERT binary.

Fill in the CONVERT questionnaire. The LAP6 manuscript is at BN 370 on unit 1.

Press line-feed.

When CONVERT is finished, type LF AP 370, CR to bring the resulting LAP6-DIAL source program into the working area.

METHOD

The LAP4-6 program scans for upper case character codes in the LAP4 manuscript and then converts them to the equivalent character codes in LAP6.

DESIGN SPECIFICATIONS

The LAP4-6 manuscript consists basically of reading, scanning, and writing routines. The converted manuscript is stored starting at block 370 on tape unit 0.

COMMENTS ON THE USE OF LAP4-6 PLUS CONVERT

For various reasons, the conversion process is seldom perfect. The original LAP4 manuscript

should always be obtained and compared to the resulting LAP6-DIAL source. If possible, compare the binary code generated by LAP6-DIAL and by LAP4.

Be suspicious of any use of the LAP6-DIAL TEXT pseudo operator in the resulting source. This will produce packed ASCII code whereas the program probably expects to find packed LINC code.

If the instructions KBD, KBD I, or TYP (OPR 14, OPR 15, or OPR 15 I) appear at any point in the source, the program must either be operated with L8SIM or else rewritten to use the 8 mode TTY instructions.

The Manus No. Systems tape (DEC-L8-LTAC-UA) contains a large number of LINC8 manuscripts which may be obtained via LAP4-6.

Title: Binary-to-LAP6 Disassembler

Author: Conrad C. Bjerke
Laboratory Computer Facility*
University of Wisconsin

PURPOSE

BINLAP6 disassembles a binary program stored on LINCtape to an equivalent LAP6 manuscript. Locations which are referred to by the address fields of instructions are tagged. The user may specify that certain areas be converted to octal numbers or text, rather than instructions. LAP6-DIAL users must then convert the resulting LAP6 manuscript into LAP6-DIAL source using the program CONVERT.

USAGE

Load BINLAP6 from any tape unit using the LOad binary command of LAP6-DIAL. Press I/O PRESET and START 20 if necessary. The initial BINLAP6 questionnaire will appear.

Before filling in the questionnaire mount a scratch tape on unit Ø (write-enabled). Mount the tape containing the binary to be disassembled on unit 1 (write-locked). BINLAP6 writes only on blocks within the LAP6-DIAL working area, and hence a LAP6-DIAL tape with the working area Cleared may be used as a scratch tape.

The following questionnaire should now be filled in.

```
    BINARY TO LAP6
(1) ?? BLOCKS
(2) STARTING AT BLOCK ???
(3) ON UNIT ?
(4) FIRST BLOCK IS Q ?
(5) STARTING LOC = ????
(6) OCTAL EQUIVALENTS ?
(7) ID ????????
(8) TAPE ID ????
(9) DATE ????????
```

In both this display and the one below, all numbers are octal and blanks are ignored. In a given field, if illegal characters, i.e., letters in a numeric field, are entered or the value entered is unacceptable, the question marks will reappear.

The questions in the above display should be answered as follows:

---

* BINLAP6 was developed by the Laboratory Computer Facility at the University of Wisconsin Medical Center under grant FR00249-04 from the Division of Research Facilities and Resources of the National Institutes of Health.

(1) The number of blocks to be disassembled should be entered here. Zero or a blank field is unacceptable. A maximum of 8 blocks may be disassembled.

(2) The location of the binary to be disassembled should be entered here. A blank field or unit number other than 0 or 1 is unacceptable.

(3) See (2).

(4) The quarter in which the first block would be loaded is entered here. A blank field indicates quarter 0. It is assumed that succeeding blocks would load into succeeding quarters until either the block count is exhausted or quarter 7 is reached.

(5) The lowest interesting location in the binary is entered here. A blank field is equivalent to 20; a value greater than 3777_8 is unacceptable. The disassembly will start either at this location or the first location of the quarter indicated in (4), whichever is greater.

(6) If it is desired to have comments in the generated manuscript indicating the value of each symbol whenever it is used, enter "Y". Anything else causes a manuscript without such comments to be generated.

(7) Whatever is entered in these fields will be included in comment lines at the beginning and end of the generated manuscript.

(8) See (7).

(9) See (7).

After question (9) is answered, the following display will appear:

```
ENTER CONSTANT AREA
TYPE ?
0 DONE
1 TEXT
2 OCTAL
3 START OVER

FIRST LOC ????
LAST LOC ????
```

This display sets up the table of areas in the binary which are not to be disassembled as instructions. For each such area, enter its type (text or octal), starting location, and ending location. The starting location must be equal to or greater than the location to start disassembly and the ending location of the previous entry in this table; a blank field is unacceptable. The ending location must be equal to or greater than the starting location; a blank field indicates a value equal to the starting location. When the questions are successfully answered, an entry is made in the table and the display reappears. If several entries are to be made in the table, they must be entered in strictly ascending order. If the first question is answered with a "3," the table is immediately cleared and the display reappears. If the first question is answered with a "0," disassembly commences immediately.

Note that option 1 will call the LAP6 text mode. CONVERT will then utilize the LAP6-DIAL text mode and you will end up with packed ASCII instead of LINC TTY code. In order to obtain LINC TTY code in the LAP6-DIAL source, specify octal disassembly for text regions. (Regions of LINC TTY text in a binary can be identified from an alphanumeric dump obtained with TAPEDUMP).

Disassembly proceeds immediately after option 0 is selected on the second questionnaire. A LAP6 manuscript will be written on unit 0 starting at BN 370. When disassembly is complete the Teletype will print the following message:

LAP6 MANUSCRIPT STARTS AT BLOCK 370 ON UNIT

USE CONVERT TO MAKE DIAL SOURCE

and the computer will halt

Note that the LAP6 manuscript now on unit 0 cannot be handled by LAP6-DIAL in any way except via CONVERT.

Move the scratch tape to unit 1. Mount a LAP6-DIAL tape containing CONVERT on unit 0.

Start LAP6-DIAL and clear the LAP6-DIAL working area.

LOad the CONVERT binary.

Fill in the initial questionnaire to indicate that the LAP6 manuscript is at location 370 on unit 1.

Press line-feed to operate CONVERT.

When CONVERT is finished, type:

LF 370, 0 CR

This will bring the LAP6-DIAL source into the working area on unit 0.

METHOD

BINLAP6 makes two passes of its binary input. During the first pass, instructions are analyzed for memory references. Memory reference fields are:

1) The α fields of SET, DIS, and XSK instructions.

2) The second word of SET instructions.

3) In index class instructions: if $\beta \neq 0$, the β field, and if $\beta = 0$ and $i = 0$, the second word.

4) The address field of full address class instructions.

Each memory location referenced which is either an index location or within the program as defined by the starting location, first quarter, and number of blocks is entered in a table.

During the second pass, manuscript is generated. The first two lines of the manuscript will be comments:

DISASSEMBLY OF iiiiiiii FROM TAPE ttttt
ddddddd

where:

iiiiiiii	is the ID entered in the first display.
ttttt	is the TAPE ID entered in the first display.
ddddddd	is the DATE entered in the first display.

This is followed by the manuscript lines representing the binary input. Each line which is referenced by some instruction will be tagged.

At the beginning of each quarter of binary, an origin line is inserted in the manuscript. Unreferenced zeros at the beginning and end of a quarter do not generate manuscript lines.

If the binary word could be a LINC instruction, the corresponding manuscript line will contain the instruction mnemonic and whatever other fields are appropriate to that type of instruction. In an index-class instruction, a nonzero β field is treated as an address. The α field of SET, DIS, and XSK is always treated as an address. For all other instructions, bits 0-3, if not specified by the operation mnemonic, are output as an octal number. In magnetic-tape instructions, the unit number is output according to the following table:

Unit #	Output
0	
1	u
4	40
5	50

If the instruction requires two words, the next word of the binary is disassembled in an appropriate manner: The second word of a SET instruction is always considered to be an address; the second word of an index-class instruction is assumed to be an address if the i-bit is not set, a constant if the i-bit is set. The second word of a half-word instruction with the i-bit set is disassembled as text. If the second word is an address and the h-bit is set, it is output as "4/" and not included in the address proper. The second word of magnetic-tape class instructions is output as quarter number-slash-block number. In CHK and MTB, if the quarter number is 0, only the block number is output.

When symbols in address fields are output and octal equivalents have been requested, the octal value of the symbol is output as a comment on the same line. Whenever an octal number is output, if its value is greater than 3777_8, i.e., the sign bit is set, its value as a signed octal number will be output in a comment on the same line.

In a text area, each word is output as a 2-character text string. If either character of the word is 13_8, 23_8, 57_8, or 77_8, which are illegal in a manuscript, that word is output as an octal number. If the second character of the word is a blank, the text output contains only the first character.

Note that since the second word of halt word instructions with the I bit set are disassembled as TEXT, they will appear in LAP6-DIAL as TEXT. Typically the resulting octal will be incorrect, and the original octal word should be inserted in place of the LAP6-DIAL TEXT code. BINLAP6 can be used to advantage if a numerical table was used by a LAP4 or LAP6 program. Exclude the entire table from disassembly. An octal source will result which, when reassembled by LAP6-DIAL, will produce the table.

Incorrect conversions can easily be obtained if the ENTER CONSTANT AREA display is abused. Do not make mistakes when filling in this table. If you do, start over (option 3). Undesired disassembly as TEXT is the most frequent result of incorrect usage of this option. Note also that the second word of a SET instruction is disassembled as a symbolic address, if the word refers to a location within the disassembly region.

Title: L8SIMØ

L8SIMØ is the LINC-8 simulator trap processor modified for direct loading into segment 0. Section 6.4 of the LINC-8 simulator trap processor document (DEC-12-S11A-D) tells exactly what was done. In addition the source of L8SIMØ has been stripped of comments so that it will occupy a minimum number of blocks in file. L8SIMØ may be added directly to user source and it will occupy Q0-Q1 of segment 0. The user program must set the instruction trap and thereafter L8SIMØ will handle KBD, KBD I, and TYP instructions.

Title: LAP4Q+A

ABSTRACT

LAP4Q+A IS Q+ASUB from the LAP4 GUIDE system. The binary of Q+ASUB was located at BN403 of the LAP4 GUIDE system and was commonly read into Q2 by user programs. If such programs are to be run in the PDP-12, it is sometimes convenient to add the LAP4Q+A source to the user program and assemble it to be directly loaded into Q2. Naturally the program must be modified so that it does not attempt to read BN403 into Q2. The binary which results when LAP4Q+A is assembled does not differ from the binary of Q+ASUB. Note that the LAP4Q+A source resets the current location counter to 1000. The routine is intended to be run in Q2.

User programs call LAP4Q+A by a JMP 1000 followed by a list of packed LINC TTY code. Unlike QANDA, LAP4Q+A places the answers to questions at locations reserved within the original text string. Half words to be filled with answers are initially filled with 13's and answer regions are delimited with 76's. The last entry in the table must be 77 or 7777.

Control returns to the calling program at the location immediately following the end of the text list. LAP4Q+A uses index registers 15, 16, 17 and occupies all of quarter 2.

The original document is attached and contains further information on the use of LAP4Q+A.

Q & A SUBROUTINE

Title: Q & A SUB Question & Answer Subroutine PDP-12

Author: M. D. McDonald, S. R. Davisson, J. R. Cox, Jr.

ABSTRACT

This is a general subroutine used to display a page of text on the scope. Question marks may be displayed and replaced with responses from the keyboard. The information entered in this manner may be recovered by a subsequent portion of the user's program.

USAGE

Q & A SUB (binary) resides in block 403 of every tape with a GUIDE system. The calling sequence for Q & A is

```
JMP 1000
```

This instruction is then followed by a table of half words of text for display. The last entry in the table is 77 or 7777. Control returns to the location of the user's program following the 77. (See example) The table of text is most easily entered into the user's manuscript by use of the Text Meta, TX (see usage description).

Obviously, before using Q & A SUB the first time, it must be read in. This is done by a:

```
RDC
21403
```

This need only be done once, if the contents of Q2 remain unchanged.

When control returns to the user's program from Q & A SUB, data may be retrieved from the table of text where 13's had been placed by TX Meta. Data is in the form of LINC code. Upper case characters are indicated by the lower case LINC code +41.

RESTRICTIONS

Q & A SUB uses Index Registers 15, 16, 17, and occupies all of quarter 2.

FORMAT FOR TEXT

1. The end of a display line is denoted by a 12.

2. The end of a page of text is denoted by a 77 or 7777. The subroutine will return control to the location following that which contains the 77 or the 7777.

3. The maximum number of characters per line is 21_{10}.

4. Each field for keyboard entries must be bordered by 76's.

5. Each question mark displayed, i.e., position for keyboard data, is denoted by a 13 by the TX Meta. Data will be in these locations in place of the 13's when subroutine exit occurs.

6. More than one question mark can be displayed per line, and more than one line can be displayed per page.

7. Any keyboard character (upper and lower case) may be displayed or entered from the keyboard, except, for CASE, del, and EOL.

OPERATION OF Q & A SUB

Once entered, Q & A SUB does the following:

1. Displays the table of text and waits for keyboard characters.

2. Q & A SUB responds to the typing of:

a. CASE by the temporary disappearance of the display until a second character is typed. The appropriate upper case character will then be displayed along with the rest of the text.

b. DEL by deleting the current answer and restoring question marks in the question presently being answered. Striking the "del" key again will restore question marks in every question of the text.

c. EOL if all characters of any one question are not filled in, pressing EOL replaced the remaining question marks with blanks; and causes the subroutine to go to the next question, (if there is one) or return to user program.

d. Any other character causes the replacement of the corresponding question mark by that character in the table of text and on the display.

Example: The following display

```
THE SUM IS XX
DO ?
O HALT
1 READ MORE DATA
```

can be obtained by use of the TEXT Meta command in combination with Q & A SUB. In this display, "XX" will be filled with a computed result by the program. The "?" will be filled from the keyboard.

In the following example, 1S contains the sum which was obtained through previous coding.

LINE NO.	BIN. LOC.	TEXT DISPLAY	MANUSCRIPT	
			:	
100)	20		LDA	Put the sum
101)	21		1S	into text for
102)	22		BCLi	display.
103)	23		7770	
104)	24		STH	Store first
105)	25		1A+4000	character into text.
106)	26		LDA	
107)	27		1S	
110)	30		SCR3	
111)	31		BCLi	
112)	32		7770	
113)	33		STH	Store second
114)	34		1A+1	character into text.
115)	35		RDC	Read in the
116)	36		21403	Display Subroutine.
117)	37		JMP .1000	Enter Display.
120)		TX_{META}		TEXT Meta command for convenience in entering the octal codes for
120)	40	TH	4733	the keyboard characters.
121)	41	E	3014	
122)	42	SU	4650	
123)	43	M	4014	
124)	44	IS	3446	
125)	45 #1A	X	1453	
126)	46	X_{EOL}	5312	
127)	47	DO	2742	
130)	50		1414	
131)	51 #2A	qQ	7613	Beginning of question.
132)	52	q_{EOL}	7612	End of question.
133)	53	0	0014	
134)	54	HA	3324	
135)	55	LT	3747	
136)	56	EOL	1412	
137)	57	1	0114	
140)	60	RE	4530	
141)	61	AD	2427	
142)	62	M	1440	
143)	63	OR	4245	
144)	64	E	3014	
145)	65	DA	2724	
146)	66	TA	4724	
147)	67	META	7777	End of text.
150	70	LDH		

LINE NO.	BIN. LOC.	TEXT DISPLAY	MANUSCRIPT
151)	71	2A+4000	
152)	72	AZEi	Test response
153)	73	HLT	Halt if 0.
		.	
		.	Continue otherwise.
		.	
170)	#1S	0	

Note: The absence of a character in the TEXT DISPLAY column above signifies a space.

Title: DSCTABLE

Author: D. A. Overton

ABSTRACT

DSCTABLE is a list of pattern words which may be used for character display in the PDP-12. This program may be added onto the user's source at any convenient point. After placing the ASCII code for any character in the accumulator, the address of the first DSC pattern word may be obtained with the following instructions:

```
BCL  I
7700
ROL  1
ADA  I
PATERN
```

The bulk of DSCTABLE is derived from QANDA. DSCTABLE gives a display interpretation to each ASCII code 00-77. The source of this program is attached for reference.

```
DSCTABLE,1  LN=1

/DSCTABLE
/PATTERN WORDS FOR CHARACTER DISPLAY.
/CALLING METHOD:
/          [(6BIT ASCII) X 2]  +  PATERN
/                    WHERE
/          PATERN = LIST STARTING ADDRESS.
/
PATERN,  Ø1Ø1        /Ø: HORIZ LINE
         Ø1Ø1        /   ILLEGAL FOR DIAL
/ALPHABETICAL CHARACTERS
         4477        /1:  A
         7744
         5177        /2:  B
         2651
         4136        /3:  C
         2241
         4177        /4:  D
         3641
         4577        /5:  E
         4145
         4477        /6:  F
         4Ø44
         4136        /7:  G
         2645
         1Ø77        /10:  H
         771Ø
         7741        /11:  I
         ØØ41
         4142        /12:  J
         4Ø76
         1Ø77        /13:  K
         4324
         Ø177        /14:  L
         Ø3Ø1
         3Ø77        /15:  M
         773Ø
         3Ø77        /16:  N
         77Ø6
         4177        /17:  O
         7741
         4477        /20:  P
         3Ø44
         4276        /21:  Q
         Ø376
         4477        /22:  R
         3146
         5121        /23:  S
         4651
         4Ø4Ø        /24:  T
         4Ø77
         Ø177        /25:  U
         77Ø1
```

```
Ø176      /26: V
74Ø2
Ø677      /27: W
77Ø1
1463      /3Ø: X
```

```
          6314
          ØØ77Ø     /31: Y
          7ØØ7
          4543      /32: Z
          6151
/SYMBOLS
          77ØØ      /33:   [UPPER CASE K = [
          ØØ41
          3Ø4Ø      /34:   \ UPPER CASE L = \
          Ø1Ø6
          41ØØ      /35:   ] UPPER CASE M = ]
          ØØ77
          772Ø      /36: ↑
          ØØ2Ø
          16Ø4      /37: HORIZONTAL ARROW
          Ø4Ø4      /    ILLEGAL FOR DIAL
          ØØØØ      /40: SPACE
          ØØØØ
          75ØØ      /41: :
          ØØØØ
          6ØØØ      /42: "
          ØØ6Ø
          3614      /43: NUMBER SIGN
          1436      /    ILLEGAL FOR DIAL
          2735      /44: $
          6735
          1446      /45: %
          613Ø
          5166      /46: &
          Ø526
          6ØØØ      /47: SINGLE QUOTE MARK
          ØØØØ      /    ILLEGAL FOR DIAL
          36ØØ      /5Ø: (
          ØØ41
          41ØØ      /51: )
          ØØ36
          2Ø5Ø      /52: *
          ØØ5Ø
          Ø4Ø4      /53: +
          Ø437
          Ø5ØØ      /54: ,
          ØØØ6
          Ø4Ø4      /55: -
          Ø4Ø4
          Ø1ØØ      /56: .
          ØØØØ
          Ø6Ø1      /57: /
          4Ø3Ø
```

```
/NUMBERS
          4536      /6Ø: Ø
          3651
          21Ø1      /61: 1
          Ø177
          4523      /62: 2
          2151
          4122      /63: 3
          2651


DSCTABLE,3 LN=63

          2414      /64: 4
          Ø477
          5172      /65: 5
          Ø651
          15Ø6      /66: 6
          4225
          4443      /67: 7
          6Ø5Ø
          5126      /70: 8
          2651
          5122      /71: 9
          3651
/SYMBOLS
          22ØØ      /72: :
          ØØØØ
          46Ø1      /73: ;
          ØØØØ
          241Ø      /74: <
          ØØ42
          1212      /75: =
          1212
          42ØØ      /76: >
          1Ø24
          4Ø2Ø      /77: ?
          2Ø55
/END DSCTABLE
/D.A. OVERTON
```

Title: PARAMS Parameter Transfer Subroutine PDP-12

Author: Dennis J. Nichols
Biomedical Computing Division*, University of Wisconsin

PURPOSE

PARAMS is an interruptable subroutine for transferring parameters to and from subroutines which require the following general calling sequence:

```
JMP Subroutine
    Parameter 1
    Parameter 2
        .
        .
    Parameter n
return here
```

USAGE

Program Implementation

The manuscript of PARAMS must be added to the users program. Only one copy of PARAMS is needed regardless of the number of routines which use it.

Manuscript implementation

PARAMS uses Index Registers 1, 2, and 3 without restoring them. The entry point is defined by the equality shown below. Since PARAMS does not reference P8, the entry point may be renamed simply by replacing the left-hand side of the equality.

```
P8 = .
```

Usage requirements

Shown below is the PARAMS calling sequence, here used by a subroutine for transferring parameters from the main program to addresses specified in the PARAMS calling sequence. This call should be the first set of instructions executed after entering the subroutine.

* Development of PARAMS was supported by grant FR00249-04 from the Division of Research Facilities and Resources of the National Institutes of Health.

```
LDA                          /LOAD CONTENTS OF REG. 0
0
JMP  P8                      /JUMP TO PARAMS
     (NT + 1) . 100₈ + NP
      ADDRESS for storing RETURN JUMP
      ADDRESS for storing PARAMETER 1
      ADDRESS for storing PARAMETER 2
            .
            .
      ADDRESS for storing PARAMETER NT
return here from PARAMS
```

NT is the number of parameters to be transferred.

NP is the total number of parameters expected by the called routine.

This value is used to calculate the return jump to the main program.

The word directly after the JMP to PARAMS must contain $(NT+1)\cdot 100_8+NP$. The second word must give the address where the return jump to the calling program is to be stored. The next NT words must contain the address for the destinations of the transferred parameters.

When PARAMS returns Index Register 1 contains 4000_8 plus the address of parameter NT+1 (although the parameter might not exist if NT = NP). The accumulator will contain the contents of the location specified by Index Register 1.

The call to PARAMS will usually occur in the following context. Note that there are three distinct levels.

MAIN PROGRAM	:
	JMP to SUBROUTINE
	a list of parameters to be used by SUBROUTINE
	:
SUBROUTINE	call to PARAMS
	rest of SUBROUTINE
PARAMS	Transfer of parameters from MAIN PROGRAM to SUBROUTINE

Space Required

PARAMS requires 36_8 locations, all in the instruction field.

Timing

LINC-8: $(88.5 + 31.5\cdot NT)$ usec.

Caution to Users

Locations 1, 2, and 3 should not normally be used as addresses for parameters to be stored.

However, location 2 may be specified as the address for the last parameter to be transferred.

Sample Programs

The following sample programs are presented as suggestions for possible ways of using PARAMS. Consider a subroutine called 1M which calculates $(x^2 + y^2)/2$ and returns the result in the accumulator:

```
                    :
                    :
MAIN PROGRAM        JMP  M1          /JUMP TO SUBROUTINE
                         X           /VALUE OF X
                         Y           /VALUE OF Y
```

One method of writing the subroutine is to transfer all the parameters.

```
SUBROUTINE  M1,     LDA              /LOAD REG. Ø
                    0
                    JMP  P8          /CALL PARAMS
                     0302            /TRANSFER 2 OUT OF 2 PARAMETERS
                     R1              /ADDRESS TO STORE RETURN JUMP
                     X1              /ADDRESS TO STORE X
                     Y1              /ADDRESS TO STORE Y
                    LDA I            /FORM X*X
            X1,     Ø                /WILL CONTAIN X
                    MUL
                    X1
                    STC  T1          /SAVE RESULT
                    ADD Y1           /FORM Y*Y
                    MUL I
            Y1,     Ø                /WILL CONTAIN Y
                    ADA I            /ADD X*X
            T1,     Ø
                    SCR 1            /DIVIDE BY 2
            R1,     Ø                /WILL CONTAIN RETURN JUMP
```

An alternate method of coding would be not to transfer either parameter as follows:

```
SUBROUTINE  M1,     LDA              /LOAD REG. 0
                    0
                    JMP  P8          /CALL PARAMS
                     0102            /TRANSFER 0 OUT OF 2 PARAMETERS
                     R1              /ADDRESS FOR RETURN JUMP
                    MUL 1            /RETURNS WITH X IN AC    FORM X*X
                    STC T1           /SAVE X*X
                    LDA  I 1         /LOAD Y
                    MUL 1            /FORM Y*Y
                    ADA  I           /ADD X*X
            T1,     Ø
                    SCR  1           /DIVIDE BY 2
            R1      Ø                /WILL CONTAIN RETURN JUMP
```

Consider a subroutine called S1 which searches for the occurrence of two characters. The subroutine starts scanning from a given address until it finds the first specified character, saves the address of that character, searches until it finds the second character, saves the address of that character and returns.

```
                    :
                    :
MAIN PROGRAM        JMP S1
                      B1            /ADDRESS OF BUFFER
                    TEXT "AZ"       /SPECIFIED CHARACTERS
                    Ø               /ADDRESS AT WHICH  A  WAS FOUND
                    Ø               /ADDRESS AT WHICH  Z  WAS FOUND
                  returns here
                    :
                  B1,  TEXT "IDEA WAS HAZY"
                    :
                    :
SUBROUTINE S1,      LDA             /SAVE REG. 0
                    0
                    JMP P8          /CALL PARAMS
                     0204           /TRANSFER 1 OUT OF 4 PARAMETERS
                     R1             /ADDRESS FOR STORING RETURN JUMP
                     2              /ADDRESS FOR STORING BUFFER ADDRESS
                    STC T1          /RETURNS WITH CHS IN AC,  SAVE FOR FIRST TEST
                    LDH 1           /RELOAD SECOND CHARACTER
                    STC T2          /SAVE FOR SECOND TEST
                    LDH 2           /LOAD FIRST CHARACTER FROM BUFFER
                    JMP .+2
                    LDH I 2         /LOAD NEXT CHARACTER
                    SHD I
                 T1, Ø
                    JMP .+2         /YES
                    JMP .-4         /NO: LOOK AGAIN
                    LDA             /LOAD ADDRESS
                    2
                    STA I 1         /STORE BACK IN MAIN PROGRAM
                    LDH I 2         /LOOK FOR LAST CHARACTER
                    SAE I           /FOUND?
                 T2, Ø
                    JMP .-3         /NO: LOOK AGAIN
                    LDA             /YES: LOAD ADDRESS
                    2
                    STA I 1         /STORE BACK IN MAIN PROGRAM
                 R1, JMP Ø          /RETURN TO MAIN PROGRAM
```

Title: DIVSUB Divide Subroutine PDP-1/2

Author: Mark Stein

ABSTRACT

DIVSUB divides a twelve bit number (eleven bits and sign) into a 24 bit number (twenty-three bits and sign). A signed eleven bit quotient and an eleven bit positive remainder, if there is one, are accessible upon subroutine return.

SPECIFICATIONS

DIVSUB uses tags: Q3, Q5, Q6, Q7 and requires the use of the tags: Q1, Q2, Q4 in the user's program.

DIVSUB uses index register 1.

Calling sequence: JMP Q3.

DIVSUB halts if divisor = 0.

Number Instructions = $65_{10} = 101_8$.

Timing: 660 microseconds.

USAGE

1) Add the manuscript DIVSUB to your program and assemble it with your program.

2) Define locations Q1, Q2, Q4 in your program. (See example)

Location	Contents when enter DIVSUB	Contents when return from DIVSUB
Q1	High Order Dividend	Remainder
Q2	Low Order Dividend	Quotient
Q4	Divisor	Two's complement negative divisor

3) Preceding each divide, put the above arguments in the locations specified above. Then JMP Q3. DIVSUB returns control to the location following the JMP Q3.

Example: To divide +3,400,000 by -3570 and put quotient in Accumulator and the remainder in S3.

MS Line No.		MS Line	Comment
0001		LDA I	
0002		0340	
0003		STC Q1	/high order dividend
0004		STC Q2	/low order dividend
0005		LDA I	
0006		4207	/-3570
0007		STC Q4	/divisor
0010		JMP Q3	/go to DIVSUB
0011		LDA	
0012		Q1	/remainder
0013		STC S3	/remainder
0014		ADD Q2	/quotient in AC
0015		HLT	
0016	#S3,	Ø	
0017	#Q1,	Ø	/argument 1
0020	#Q2,	Ø	/argument 2
0021	#Q4,	Ø	/argument 3

Add the DIVSUB source and assemble

4) Restrictions on arguments:

a. Divisor: $\pm$ 3777_8 and not equal to zero. DIVSUB halts if the divisor = 0. Pressing resume will continue the division, but, of course, the quotient and remainder will be erroneous.

b. Dividend: $\pm$ 37777777

c. The Quotient will always be between -3777 and +3777. Therefore, division which should yield a quotient larger than this is erroneous.

For example: dividing 100,000 by 1 will give an erroneous result.

To insure against this: for each number of eleven signed bits or less, added to form the dividend, a number greater than 0 must be added to form the divisor.

METHOD

The sign of the quotient is determined first. Then twelve successive subtracts are performed to form the quotient and remainder.

This subroutine does not make use of the MQ register.

Title: SPFLT Single Precision Floating Point Package

Authors: J. C. Dill, W. M. Stauffer, R. W. Stacy
Biomathematics Laboratory, University of North Carolina, Raleigh, North Carolina

ABSTRACT

The floating point package provides single precision addition, subtraction, multiplication, and division. Also contained are subroutines for transmission and for fix-to-float conversion.

The package occupies all of Q3.

INTRODUCTION

We are indebted to James Bryan of NIH for some of the ideas used in these programs.

USAGE

1. Standard form of single precision floating point number.

a.

	Binary Representation	
M2,	+XX XXX XXX XXX	mantissa
	±XX XXX XXX XXX	exponent

b. Binary floating point numbers. The left-most bit is the sign bit, and the most significant bit is the first bit to the right of the sign bit. The binary point is assumed to be between the sign bit and the most significant bit.

Example:

$$\frac{152_{10}}{19_{10}} = \frac{0.10\ 011\ 000\ 000_2 \times 2^{1000_2}}{0.10\ 011\ 000\ 000_2 \times 2^{101_2}}$$

$$= 0.10\ 000\ 000\ 000 \times 2^{100_2}$$

$$= 8_{10}$$

To perform this division:

loc.	contents	
20	JMP DIVSP	
21	100	
22	102	
23	104	
24	HLT	
.		
.		
.		
.		
.		
100	010 011 000 000	dividend
101	000 000 001 000	
102	010 011 000 000	divisor
103	000 000 000 101	
104	010 000 000 000	quotient (initially,
105	000 000 000 100	these two locations may have any value)

c. The exponent must be in the location following the mantissa.

d. Both the mantissa and the exponent are signed eleven bit numbers.

e. Floating point numbers are always addressed by the mantissa.

2. Use of the floating point subroutines.

a. Preparation. SPFLT may be converted as part of your own program if your program does not contain any of the tags SPFLT uses (See restrictions).

b. Calling Sequence and Arguments.

Entrances

The entrance to the various operations are:

SUBTRACT---------- SUBSP (1437)
ADD---------- ADDSP (1446)
DIVIDE----------DIVSP (1565)
MULTIPLY---------- MULSP (1671)
TRANSMIT FLOATING---------- TRANS (1771)
FIX-TO-FLOAT----------FIXFL1 (1726)
FIX-TO-FLOAT----------FIXFL2 (1730)

C(AC) = address of number - FIXFL1
C(AC) = number---------- FIXFL2

Arithmetic Instructions

1. The accumulator need not be cleared before jumping to a floating operation, and will be clear after the jump back to the main program. Anything in the accumulator when the jump is made, however, will be destroyed.

2. To perform the floating operation A + B, A - B, A x B, or A/B, the following instructions are used:

loc.	contents	
.	JMP X	where X is the entrance to
.+1 α		the appropriate floating
.+2 β		operation, α and β are the
.+3 γ		addresses of A and B, and
.+4	next program command	γ is the address desired for the result.

Note that for division and subtraction, the order of the operand addresses is important. On completion of the operation, a jump is made to location .+4.

Transmit Floating Instruction

1. This instruction moves a floating point number from one address to another. The conditions are the same as for the arithmetic instructions.

2. Example:

loc.	contents
.	JMP TRANS
.+1	α
.+2	β
.+3	

The floating point number at address α is moved to address β. The contents of α and α+1 are unchanged. Control returns to location p+3.

Fix-to-float Convert Instruction

1. Index Register 3 must be set to the desired address of the floating point mantissa. Register 3 will be indexed by one by the subroutine.

2. Load either the number to be converted, or its address, into the accumulator, and JMP FIXFL1 (1730) if A = number, JMP FIXFL2 (1726) if A = address.

3. If a long list of numbers is to be converted, the following change in the routine is recommended:

loc.	present contents	recommended change
1726	STC 1	NOP
1727	LDA 1	LDA I 4

The format for conversion would then be:
SET I 4
1st fixed point address -1
SET I 3
desired address of 1st floating point number.
JMP 1726
XSK I 3

loc.	present contents	recommended change

ADD 4, or ADD 3
last fixed point address, or last floating point address +2.
JMP .-5
In this case, both register 3 and 4 will be indexed by one by the subroutine, and 3 will be indexed by 1 by the main program.

SPECIAL FEATURES

1. Rounding - There is an internal subroutine which will round the number in the accumulator according to the absolute value of the Link bit. This subroutine has two entrances R1 and R2. A jump to R2 will always round up if the absolute value of the Link bit is 1, and not otherwise. A jump to R1 will round up half the time and down half the time if the absolute value of the Link bit is 1, and not otherwise.

2. Zero - If the result of any floating operation has a zero mantissa, the exponent is set to zero.

3. Rounding in addition and subtraction -

a. If the smaller number is to be scaled, A is always rounded up if the absolute value of the last bit scaled off is 1.

b. Scaling and rounding are avoided if:

1) Exponents are equal .
2) Number with larger exponent is zero.

4. Multiplication and Division

a. In division, if the numerator must be scaled (to make numerator < denominator) it is rounded up half the time if the absolute value of the scaled bit is 1.

b. Division by zero always produces a number with an exponent of 3700_8. This is arbitrary, and is easily changed. (Location 1577)

c. If a floating multiply or divide produces an exponent whose absolute value is > 3716, the program halts in the exponent check routine at location 1767. This HLT is arbitrary and may be removed by changing location 1764.

d. A 12 bit product (quotient) is always formed. If the product (quotient) as formed is not normalized, it goes to a normalize routine. If the result is normalized, it is rounded up half the time.

All results of floating operations are left in normalized form.

METHOD OF DIVIDE

Since we are not using the standard algorithm for divide, we decided to include a short explanation of the method used. It is adapted from a method described in Chapter 15, R. S. Ledley, Digital Computers and Control Engineering, McGraw-Hill Book Company, Inc., 1960, 835 pp.

Let N = numerator
D = denominator
Q = quotient

Assume $0 \leq N < 1$, $0 \leq D < 1$, $N < D$

Then $N/D = Q = Q_1 2^{-1} + Q_2 2^{-2} \ldots + Q_n 2^{-n}$

Form $A = 2N - D = D(Q_1 2^{0} + Q_2 2^{-1} + Q_3 2^{-2} \ldots + Q_n 2^{-(n-1)})$

Since we assumed $N < D$, Q_1 is 0 or 1 depending on whether A is greater or less than zero. The remaining digits may be found in a similar manner. The method is flow-charted below.

RESTRICTIONS

SPFLT uses Index Registers 1, 2, 3. It occupies 400 locations in the instruction field, normally Q3, and uses the following symbolic addresses:

A1, A2, A4
D1, D2, D3, D4, D5, D7
E1, E2, E3
K1, K2, K3
M1, M2, M3, M4
N1, N2
P1, P2, P3
R1, R2, R3
Z4, Z5

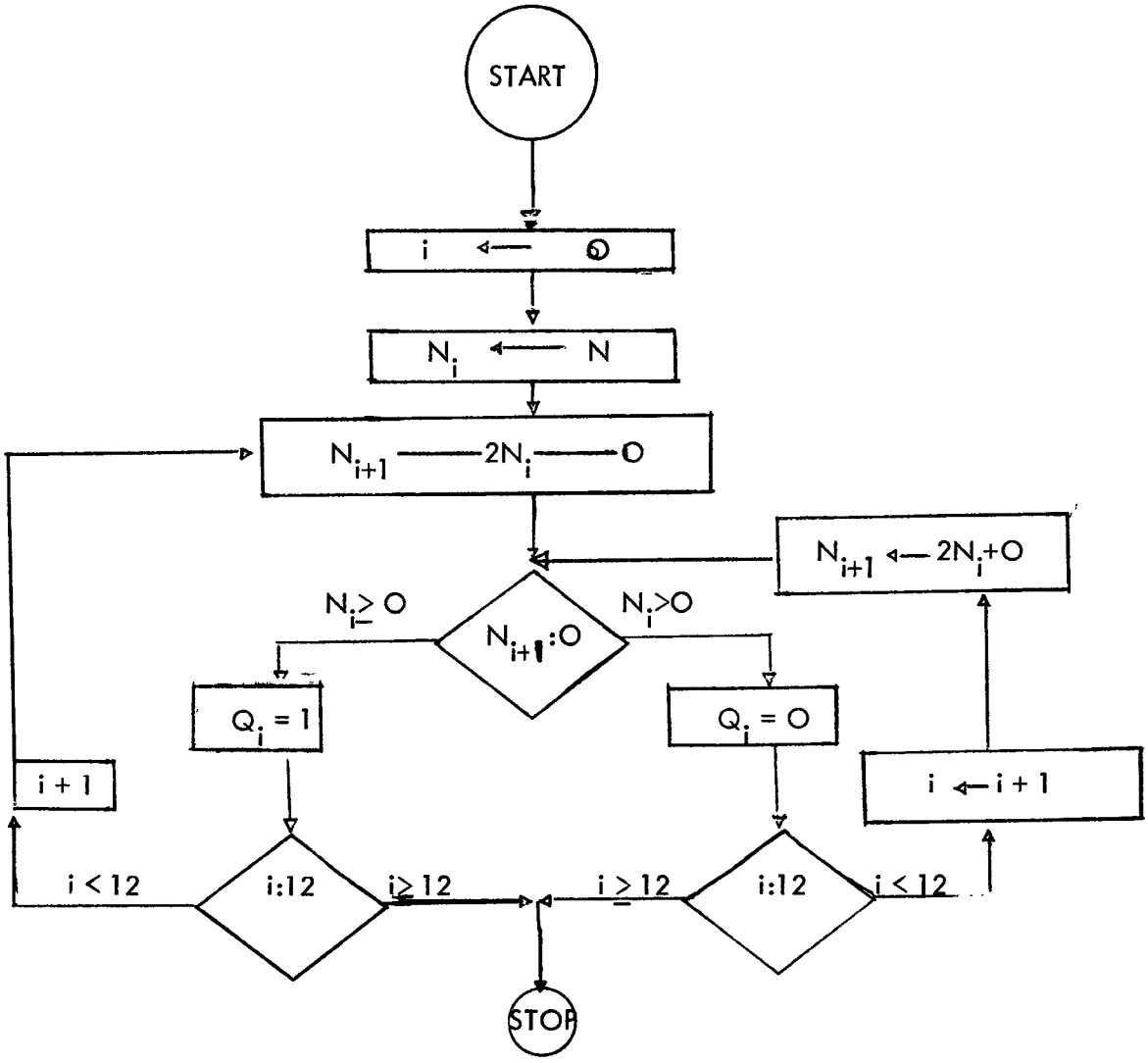
SUBSP, ADDSP, DIVSP, MULSP, TRANS, FIXFL1, FIXFL2

PROGRAM TIMING

Addition	Subtraction	Multiplication	Division
0.2 msec.	0.2 msec	0.28 msec	0.85 msec

These times are approximations and should be used as such.

DIVIDE FLOW CHART



Title: QIKDIV Quick and Dirty Integer Divide PDP-12

Author: James Hance

ABSTRACT

This program (subroutine) provides a very fast division, with limited accuracy, for positive integers of less than 50_8 (40_{10}). This is useful for scaling displays, simple averaging programs, and similar situations where the loss of accuracy and the limitations imposed by this subroutine can be tolerated.

USAGE

Upon entry to this subroutine, the dividend (a signed 11 bit number) must be in the accumulator, and the positive integer divisor ($0-50_8$) must be in an address tagged A7 within the calling program.

Upon exit from QIKDIV, the divisor will be untouched in A7 and the accumulator will contain the quotient. The remainder is lost; there is no rounding of the quotient.

Negative or zero divisors will result in a rapid exit from the subroutine with the dividend unchanged in the accumulator. Divisors greater than 50_8 will be assumed to be 50_8.

SPECIFICATIONS

Number of locations	75_8
Index Registers	none
Tags	D7 in QIKDIV A7 in calling program
Calling Sequence	A7, Ø LDA I 2 /divisor STC A7 ADA I 3 /dividend JMP D7

Times: 73.5usec (30 or 33 usec if divisor is negative or zero respectively).

Limitations: Divisor must be - -
positive integer
between $1-50_8$.

Accuracy: The worst case occurs when divisor is 50_8. Error may reach 5% maximum.

NOTE: This is a LAP4 Listing

```
                0001        [QIKDIV
                0002        [QUICK AND
                0003        [DIRTY DIVISION
                0004        [SUBROUTINE
                0005        [AJH JAN 65
0020            0006          #7D  STC p+23        /save dividend
0021            0007               ADD O
0022            0010               STC 1+17       /save return jump
0023            0011               ADD 7A         /load divisor
0024            0012               APOi
0025            0013               AZEi           /fast exit provided if divisor
0026            0014               JMP p+14         is negative or zero
0027            0015               ADAi
0030            0016               - 50           /check if divisor exceeds 50(8)
0031            0017               APOi
0032            0020               CLR            /if divisor exceeds 50(8), set it
0033            0021               ADAi             to equal 50(8). In either case
0034            0022               p+2060           use this to form an ADD instruction,
0035            0023               STC p+1          used below
0036            0024               ADD 000        /this loads the reciprocal of the
0037            0025               MUL              divisor from the list
0040            0026               p+4003         /fractionally multiply reciprocal
0041            0027               JMP 000          with dividend and jump back to
0042            0030               LDAi             calling program
0043            0031               0              /before fast exit (divisor - or 0)
0044            0032               JMP p-3        restore dividend to accumulator
0045            0033               3777           /reciprocal of 1
0046            0034               2000           /reciprocal of 2
0047            0035               1252           /reciprocal of 3
0050            0036               1000           /reciprocal of 4
0051            0037               631            /reciprocal of 5
0052            0040               525            /reciprocal of 6
0053            0041               444            /reciprocal of 7
0054            0042               400            /reciprocal of 10
0055            0043               343            /reciprocal of 11
0056            0044               314            /reciprocal of 12
0057            0045               272            /reciprocal of 13
0060            0046               252            /reciprocal of 14
0061            0047               235            /reciprocal of 15
0062            0050               222            /reciprocal of 16
0063            0051               210            /reciprocal of 17
0064            0052               200            /reciprocal of 20
0065            0053               170            /reciprocal of 21
0066            0054               161            /reciprocal of 22
0067            0055               153            /reciprocal of 23
0070            0056               146            /reciprocal of 24
0071            0057               141            /reciprocal of 25
0072            0060               135            /reciprocal of 26
0073            0061               131            /reciprocal of 27
0074            0062               125            /reciprocal of 30
0075            0063               121            /reciprocal of 31
0076            0064               116            /reciprocal of 32
```

```
0077    0065            113     /reciprocal of 33
0100    0066            111     /reciprocal of 34
0101    0067            106     /reciprocal of 35
0102    0070            104     /reciprocal of 36
0103    0071            102     /reciprocal of 37
0104    0072            100     /reciprocal of 40
0105    0073            76      /reciprocal of 41
0106    0074            74      /reciprocal of 42
0107    0075            72      /reciprocal of 43
0110    0076            70      /reciprocal of 44
0111    0077            67      /reciprocal of 45
0112    0100            65      /reciprocal of 46
0113    0101            64      /reciprocal of 47
0114    0102            63      /reciprocal of 50
        0103    [END
```

Title: RANDOM

Author: Mark F. Lewis, W. Madeline Webber, Jimmie E. Doyle

ABSTRACT

The random number subroutine is a translation into LINC language of the DECUS Program No. 5-25, "A Pseudo Random Number Generator for the PDP-5 Computer," by P. T. Brady. The subroutine simulates on the LINC the 2's complement arithmetic of the PDP-5 subroutine, but returns with the equivalent LINC number in the accumulator.

USAGE

Load the subroutine in any instruction field. To enter the subroutine:

```
LDA I
 N          where N = octal # of IF where user's main program resides

LIF X       where X = IF containing random number generator

JMP RANDOM
```

Control returns to the user's program at the next location with the random number in accumulator.

Random occupies 222 locations. Time less than 0.5 m sec.

METHOD

See: Brady, P. T., "A Pseudo-Random Number Generator for the PDP-5 Computer," DECUS No. 5-25.

SEQUENCES OF LONG RUNS

After use, a core image of the subroutine can be written over the original routine on LINCtape. In this way, the subroutine can generate over 4 billion numbers without repeating the sequence and without use of the initialization suggested by Brady.

Title: NU INDEX

USAGE

This LAP6-DIAL source makes it possible to construct any desired LAP6-DIAL index, thereby placing programs at any location which you may select within the LAP6-DIAL file area. First the NU INDEX source is modified so that it will produce the desired index. Then the resulting binary is copied into the blocks 346-7 where a LAP6-DIAL index belongs. Finally, material may be copied into the reserved regions in the LAP6-DIAL file. Note that LAP6-DIAL cannot file binary or source according to the index prepared with NU INDEX. LAP6-DIAL will simply avoid filling anything else in the regions which you have reserved.

INDEX STRUCTURE

The first eight words of each LAP6-DIAL index block are filled with the digits 5757. This identifies the index block. Each succeeding group of eight words defines one index entry. The first four octal words define the program name. The last four octal words define the size and location of the binary and source file entries for the program. The source for NU INDEX defines two full blocks of binary - a complete LAP6-DIAL index. Except as you modify the source, all locations in these two blocks will be 5757. You may modify as many chunks of 8 words each as you wish to define index entries. Be certain that you replace as many locations as you delete so that you end up defining two full binary blocks. Note that when the binary generated by NU INDEX is saved, a three-block binary index entry will result. The first block is a header block; the next two blocks are the LAP6-DIAL index binary.

PROGRAM NAME

The program name is entered in 6-bit ASCII characters. A maximum of 8 characters may be used, thereby filling four words. Unused characters at the end of a shorter name must be filled with 77. If the TEXT mode of LAP6-DIAL is used to generate the ASCII code for these four words, the number 77 may be produced by the symbol "?."

BINARY AND SOURCE LOCATIONS

Within each block of 8 octal words, the fifth word indicates the starting block number of the source manuscript. The sixth location indicates the number of blocks in the source manuscript. The seventh location indicates the starting block number of the program binary. The eighth location indicates the number of blocks used by the binary. All four of these numbers are specified in octal. The maximum block number is 777. If there is no source for a given program, locations 5 and 6 must be filled with numerals 5757. If there is no binary, locations 7 and 8 must be filled with 5757.

Title: QANDAS

QANDAS is identical to QANDA except that all comments have been omitted to reduce the number of blocks occupied by the source. It may be assembled and used exactly as described in the QANDA document (DEC-12-FISA-D).