HDL-TM-91-12

August 1991

A PC Program to Calculate the One-Sided Lower Tolerance Limit for Total Ionizing Dose Radiation Data

by Ahmed A. Abou-Auf
Mark E. Bumbaugh

U.S. Army Laboratory Command
Harry Diamond Laboratories
Adelphi, MD 20783-1197

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE August 1991	3. REPORT TYPE AND DATES COVERED Final, from Sept 1990 to Jan 1991

4. TITLE AND SUBTITLE A PC Program to Calculate the One-Sided Lower Tolerance Limit for Total Ionizing Dose Radiation Data	5. FUNDING NUMBERS PE: 62120
6. AUTHOR(S) Ahmed A. Abou-Auf and Mark E. Bumbaugh	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Harry Diamond Laboratories 2800 Powder Mill Road Adelphi, MD 20783-1197	8. PERFORMING ORGANIZATION REPORT NUMBER HDL-TM-91-12
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Laboratory Command 2800 Powder Mill Road Adelphi, MD 20783-1145	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES AMS code: 612120.H250011 HDL PR: 1XE724	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The radiation failure level of a semiconductor part type can be found to be just above the level needed for survivability. Many times the part has to be accepted or rejected lot to lot (lot qualification) on the basis of radiation data from a small sample of the lot. One-sided lower-tolerance-limit statistics is used on the radiation data to determine lot acceptance or rejection. An IBM-compatible PC program, written to calculate the one-sided lower tolerance limit at 90- to 50-percent confidence for log-normally distributed radiation data (the usual distribution for semiconductor radiation data), is described in this report. The analysis results can be displayed in tabular and graphical form on the PC screen and a hard copy made on the printer.

14. SUBJECT TERMS Piecepart radiation data, hardness assurance, lot qualification, one-sided lower tolerance limit, log-normal distribution, personal computer, C language, printer control language	15. NUMBER OF PAGES 61
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	17. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT SAR

Contents

Executive Summary

This report describes a software program developed by Harry Diamond Laboratories (HDL) to analyze piecepart radiation data for hardness assurance purposes. It can be used for piecepart lot acceptance based on a small test sample size. The program was written because no user-friendly program was discovered that had easy data entry and multiple output formats. The program calculates the one-sided lower tolerance limit at both 90- and 50-percent confidence for radiation data that are log-normally distributed (the usual distribution for semiconductor parts). The software runs on an IBM or compatible XT, AT, or 386 personal computer (PC). The results can be displayed in tabular or graph form on the PC screen, and a hard copy can be made on any printer that uses the Printer Control Language (PCL). The program is written in the C language and compiled with a Turbo C compiler; it can be copied and modified to suit one's purposes.

1. Introduction

If the radiation failure level of a semiconductor part is near the level needed for survivability then lot acceptance testing must be done based on radiation data from a small sample of the lot. To determine whether a lot is acceptable, one-sided lower-tolerance-limit statistics are used on the radiation data, which are assumed to be log-normally distributed. Since this type of statistics is not in common use, commercial software packages do not exist. Some programs have been developed by those working in the radiation testing area, but they are either cumbersome or not user friendly. Harry Diamond Laboratories (HDL) has written a program for analyzing lot sample radiation data that is user friendly and has easy data entry and multiple output formats.

The purposes of this software program are to make the personal computer (PC) a tool in analyzing piecepart data for nuclear radiation hardness assurance purposes and to facilitate such data analysis, especially for small sample sizes. Because of the time and cost of radiation testing, data are usually obtained on a limited number of parts. If large variations exist in the failure levels of a part type, or if the failure levels are not significantly above that needed for survivability (for example, if a part's failure levels are less than five times the system survival level), then lot-qualification is often used. This qualification is done by testing a small number of parts from the same lot and using statistical analysis to either qualify or reject the lot. Also, the same statistical analyses are used on multiple diffusion lots to qualify part types for hardness assurance at radiation levels M, D, R, and H in MIL-M-38510 and MIL-S-19500.

Since the evaluation of radiation test data by different industry and government facilities indicates that most radiation data have a log-normal distribution,[1] our software program assumes that the data entered are log-normal. Therefore, this program should only be used with data that have an approximate log-normal distribution. For multilot data or other cases where the distribution is not well-behaved, the reader should use the techniques discussed in MIL-HDBK-816, *Guidelines for Developing Radiation Hardness Assured Device Specifications*. The program calculates the probability of failure as a function of the radiation environment using one-sided lower tolerance limit (OSLTL) statistics, assuming that the failure levels entered have a log-normal distribution. The calculations are performed for two confi-

[1]MIL-HDBK-279, *Total-Dose Hardness Assurance Guidelines for Semiconductor Devices and Microcicuits* (25 January 1985), section 5.1.3, 12.

dence levels: 50 and 90 percent. The OSLTL with 90-percent confidence is the statistical approach usually recommended for radiation hardness assurance.[1,2] For example, many times when radiation data are taken on a part lot having marginal failure levels, the calculated radiation level at the 1-percent probability-of-failure (99-percent probability of survival) point (using OSLTL statistics at 90-percent confidence) has to equal or exceed the system's survival level in order to be a qualified lot.

The calculated results of the program can be viewed on the PC screen or printed out in either table or graph form, and each form also lists the raw data on the printout but not on the screen display. The table lists the radiation levels corresponding to 19 values of the probability of failure (1%, 5%, 10%, 15%, ... 90%). See figure 1 as an example of a tabular printout. The graph is a curve of the probability of failure versus the radiation level. Two curves are drawn: one for 90-percent confidence and one for 50-percent. Also plotted on the graph are the

Figure 1. Tabular printout (option 3).

```
           One Sided Tolerance Limit Assuming a Log Normal Distribution


           Prob.           50% Conf.           90% Conf.
           of failure
           1.0             4.97E+03            2.74E+03
           5.0             7.19E+03            4.55E+03
           10.0            8.71E+03            5.90E+03
           15.0            9.90E+03            6.99E+03
           20.0            1.10E+04            8.03E+03
           25.0            1.21E+04            9.01E+03
           30.0            1.31E+04            9.95E+03
           35.0            1.40E+04            1.08E+04
           40.0            1.51E+04            1.18E+04
           45.0            1.61E+04            1.27E+04
           50.0            1.73E+04            1.38E+04
           55.0            1.85E+04            1.48E+04
           60.0            1.97E+04            1.59E+04
           65.0            2.12E+04            1.71E+04
           70.0            2.28E+04            1.83E+04
           75.0            2.47E+04            1.98E+04
           80.0            2.70E+04            2.15E+04
           85.0            3.01E+04            2.37E+04
           90.0            3.42E+04            2.66E+04

Caution:
  Prob. of failure taken below 9.1 or above 90.9 could be inaccurate.

           Dev.   Fail. Level
           No.    rads (Si)
           1      7.30E+03
           2      1.05E+04
           3      1.10E+04
           4      1.30E+04
           5      1.70E+04
           6      1.80E+04
           7      2.20E+04
           8      2.40E+04
           9      3.30E+04
           10     4.00E+04
```

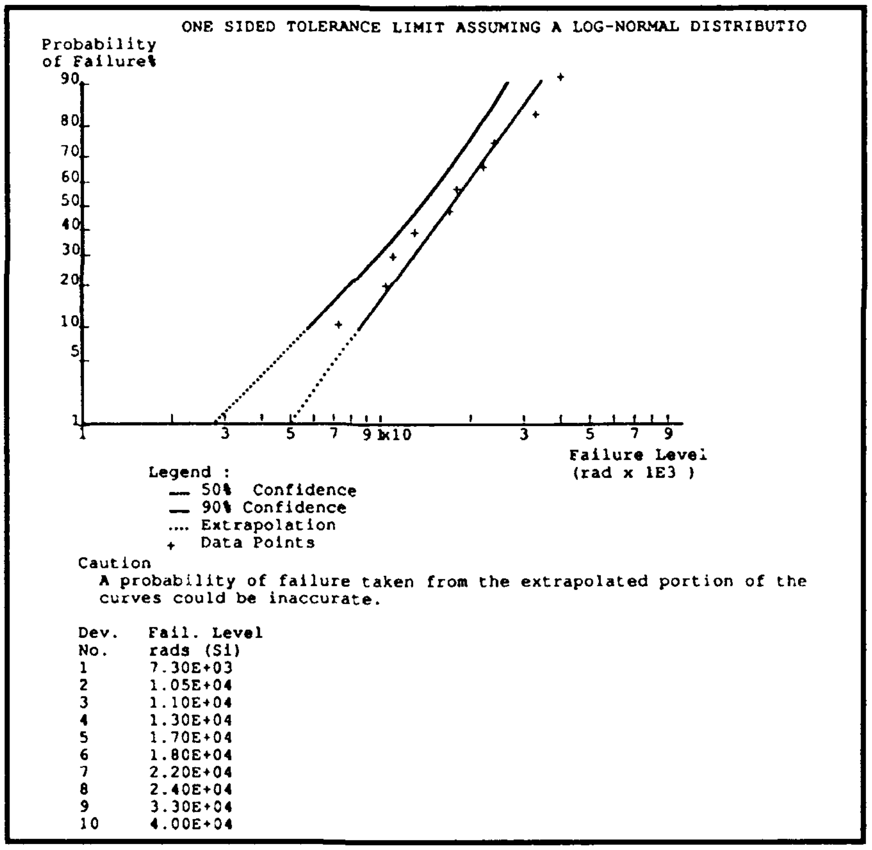
[1]MIL-HDBK-279, *Total-Dose Hardness Assurance Guidelines for Semiconductor Devices and Microcicuits* (25 January 1985), section 5.1.3, 12.
[2]I. Arimura, R. A. Kennerud, and O. R. Mulkey, *Hardness Assured Device Specification*, Defense Nuclear Agency, TR-81-90 (15 July 1982), section 5-2.2, 107-108.

raw data points entered into the program for calculating the OSLTL values. See figure 2 as an example of a graph printout.

The hardware requirement for the program is an IBM or compatible model XT, AT, or 386 PC. For obtaining a copy of the results, a printer is needed that accepts Printer Control Language (PCL), such as the HP Laserjet II, IBM Laser Printer 4019, etc. The program is written in the C language, is compiled using the Turbo C compiler version 2.0, uses less than 85K of random access memory (RAM) space for the executable file, and uses various available Turbo C language graphics driver files (ATT.BGI, CGA.BGI, EGAVGA.BGI, HERC.BGI, IBM 8514.BGI, and PC 3270.BGI), depending on the monitor used.

Experience in using PCs and a familiarity with the C programming language is necessary to understand section 4 of this report, but only PC experience is necessary to run the program.

Figure 2. Graph printout (option 5).



```
Legend :
   —— 50%  Confidence
   —— 90% Confidence
   .... Extrapolation
    +   Data Points
Caution
  A probability of failure taken from the extrapolated portion of the
  curves could be inaccurate.

Dev.   Fail. Level
No.    rads (Si)
1      7.30E+03
2      1.05E+04
3      1.10E+04
4      1.30E+04
5      1.70E+04
6      1.80E+04
7      2.20E+04
8      2.40E+04
9      3.30E+04
10     4.00E+04
```

2. One-Sided Tolerance Limit Statistics

The one-sided lower tolerance limit represents a statistically conservative method of estimating the probability of an event (percentage of probability of failure in this report) relative to an assumed distribution (log-normal for radiation-degraded semiconductors) given a number of samples (IC failure levels). For a normal distribution, the one-sided lower tolerance limit (X_L) is given by the equation[2]

$$X_L = \overline{X} - K_{TL}S \ ,$$

where the distribution mean (X) is calculated by

$$\overline{X} = 1/n \sum_{1}^{n} x_i \ ,$$

the standard deviation (S) is calculated by

$$S = \left[1/(n-1) \sum_{1}^{n} \left(x_i - \overline{X}\right)^2 \right]^{1/2} \ ,$$

and the factor for one-sided tolerance limits for a normal distribution (K_{TL}) is obtained from statistics tables. Here the letter n is the sample size, while X_i represents the value of the i^{th} sample.

The K_{TL} factor varies with the sample size (n), the percentage of probability of failure (X_L), and the percentage of confidence desired in the probability-of-failure calculations. The tables for K_{TL} factor values usually supply only the factor for probabilities at and above 75 percent. Therefore, the following equation[2] (which is most accurate at sample sizes of 10 and above*) was used to calculate K_{TL}.

$$K_{TL} = \left\{ Z_p + \left[Z_p^2 - ab \right]^{1/2} \right\} / a \ . \tag{1}$$

Here

$$a = 1 - \left(Z_\gamma^2 / 2\,(n-1) \right) \text{ and } b = Z_\gamma^2 - \left(Z_p^2 / n \right) \ , \tag{2}$$

where n is the sample size,

Z_p is the standard normal variable for the percentage of probability (P), and

Z_γ is the standard normal variable for the percentage of confidence (γ).

The values for Z_p and Z_γ can be found in "standard normal variable" tables and are reproduced in appendix A.[3] This table gives the Z value for the probability of survival. The probability of failure is simply the complement of this value. For example, from appendix A, $Z = +2.33$ for

[2] I. Arimura, R. A. Kennerud, and O. R. Mulkey, *Hardness Assured Device Specification*, Defense Nuclear Agency, TR-81-90 (15 July 1982), section 2-13/15.
[3] Mary Gibbons Natrella, *Experimental Statistics*, National Bureau of Standards Handbook 91 (1963), T-3.
*See also section 3.9 for a discussion of sample size.

99-percent probability of survival, but by subtracting 0.99 from 1.00 to get the complement, 0.01, we see that $Z = -2.33$ for 1-percent probability of failure.

Radiation damage in semiconductors usually follows a log-normal distribution. For a log-normal distribution, the logarithm (common or natural) of the failure levels has a normal distribution and, therefore, the one-sided lower tolerance limit equation (using the natural log) is

$$\ln X_L = \overline{\ln X} - K_{TL} S_{\ln} \,, \tag{3}$$

where

$$\overline{\ln X} = 1/n \sum_{1}^{n} \ln x_i \tag{4}$$

and

$$S_{\ln} = \left[1/(n-1) \sum_{1}^{n} \left(\ln x_i - \overline{\ln X}\right)^2 \right]^{1/2} \,. \tag{5}$$

With this equation, the ln X_L is calculated for the confidence and sample size at the percentage of probability of failure of interest. The antilog of the ln X_L yields X_L (the failure level associated with that probability). The one-sided tolerance limit can be calculated for different confidence levels. Only two confidence levels (90 and 50 percent) are used by the software program described in this report. Thus we calculate the radiation level above which we may predict with either 50- or 90-percent confidence that a portion of the population of ICs will lie.

You can use log-normal probability graph paper to check whether the failure levels were log-normally distributed. To do this, the n failure levels of X_i are ranked from the lowest value (X_1) to the highest (X_n). Next to the i^{th} value, write the number $i/(n + 1)$ to make a list like this:

X_1	$1/(n+1)$
X_2	$2/(n+1)$
X_3	$3/(n+1)$
	...
X_n	$n/(n+1)$

Then the values of X_i are plotted along the normal distribution axis representing the percentage of probability of failure against the corresponding $i/(n + 1)$ on the logarithmic axis representing the failure level. The plus signs in figure 2 are an example of points plotted in this manner. When failure levels represent a log-normal distribution, the points will be in an approximate straight line where the line's 50-percent probability point intercepts the approximate value of the mean (X) of the distribution, and the line has a slope of approximately the reciprocal of the standard deviation ($1/S$).

3. Instructions for Use

To use the program, set the working directory to the directory that contains both the executable program file (for example, HDL's file was named OSTL.EXE) and the graphics file (*.BGI) needed for the PC. The graphics file is the particular video driver needed based on the PC's hardware. Run the program by typing "OSTL" and pressing the enter key. The Main Menu screen will appear (see fig. 3). The Main Menu has eight options; however, you must enter the raw data first by either using the keyboard (Option 1-Enter Data) or by reading a file containing the raw data (Option 7-Retrieve File). After the raw data are entered, the other options can be used. You can usually use the backspace key anytime in this program before the enter key is pressed to erase incorrect keyboard entries.

3.1 Enter Data (Option 1)

To enter raw data from the keyboard, go to the Main Menu screen and type the number 1. A new screen will appear having the messages "Devices Number =", "Units=", and "Enter Data" (see fig. 4). Type in the number for the amount of raw data to be entered (range of 4 to 54) and press the return key. The number will appear at the top of the screen, and a new message at the bottom of the screen will request that a "1" be typed if the data to be entered are in "rads" or a "2" for units of "n/sq cm." After you enter a 1 or 2 and press the return key, a

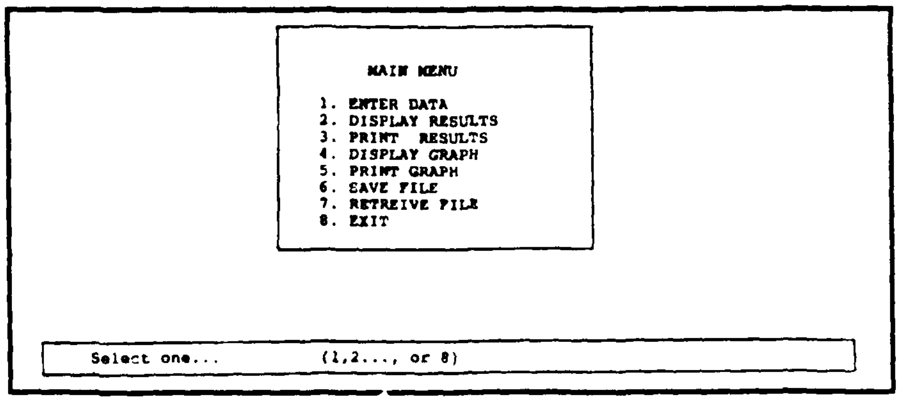
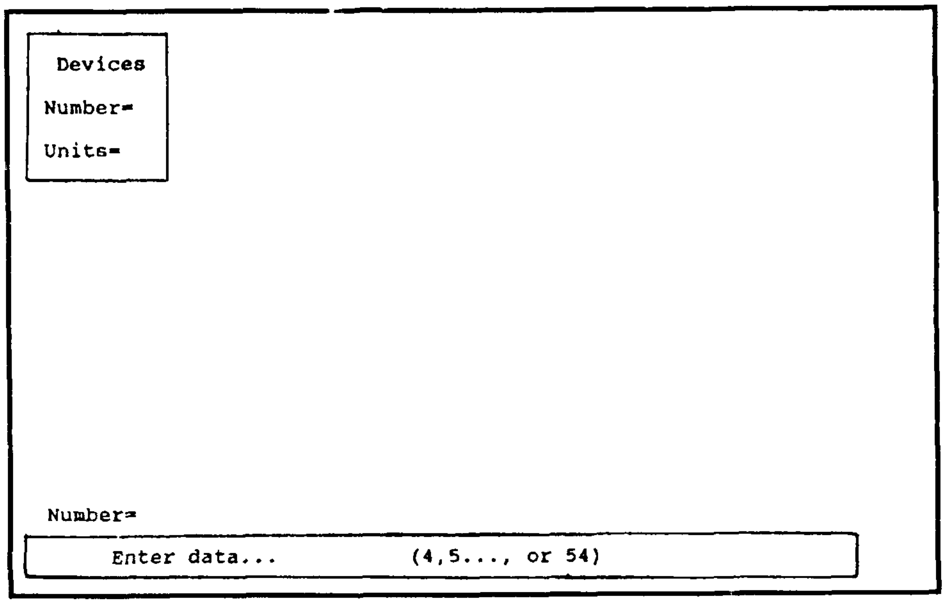


Figure 3. Main menu screen.

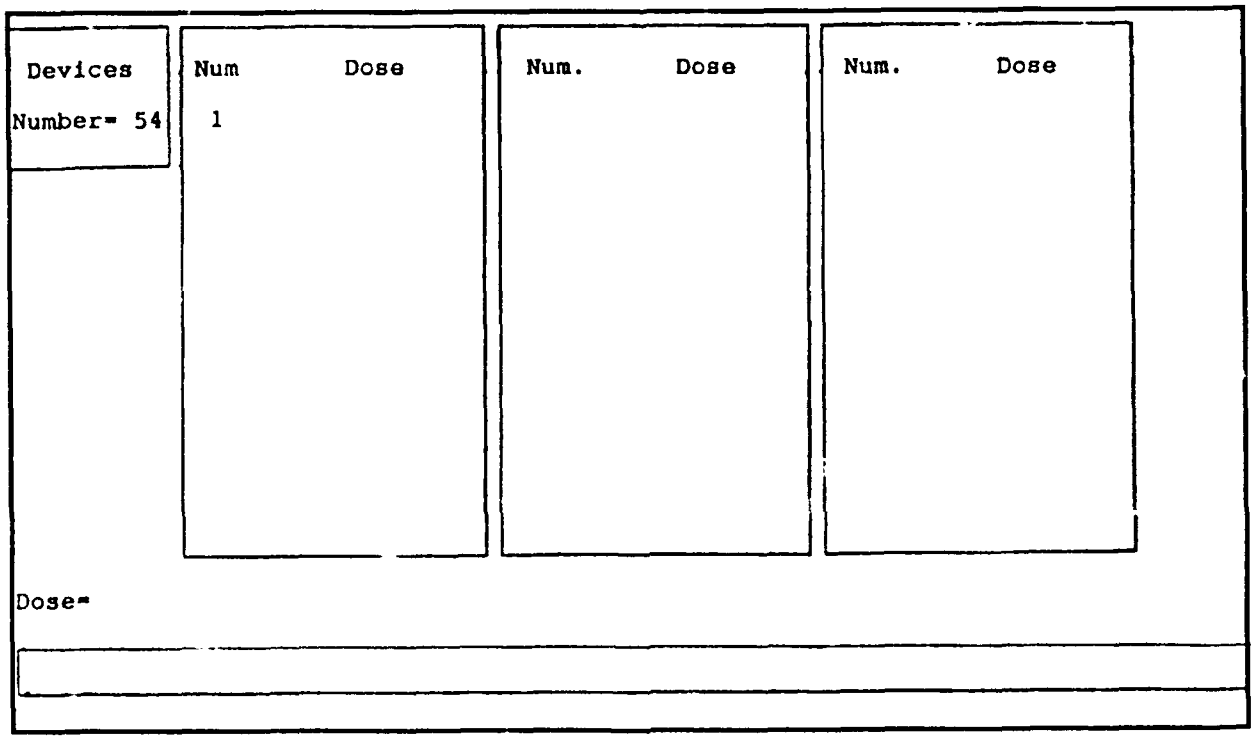
message at the bottom will ask if you want to change those numbers. If so, type "Y" and enter the new numbers. If not, type "N" and a blank table will appear on the screen (fig. 5).

Type the value for the first data point and press enter. The value can be in scientific notation (e.g., $1 \times 10^{12} \equiv$ 1E12). Entries that are specified with more than one decimal place will be rounded to one decimal place. (For example, 44.55 is rounded to 44.6). Entries greater than 9,999,999 will be given a close approximation (for example, 66,666,666

Figure 4. Devices number/enter data screen (option 1).



```
Devices
Number=
Units=

Number=
Enter data...          (4,5..., or 54)
```



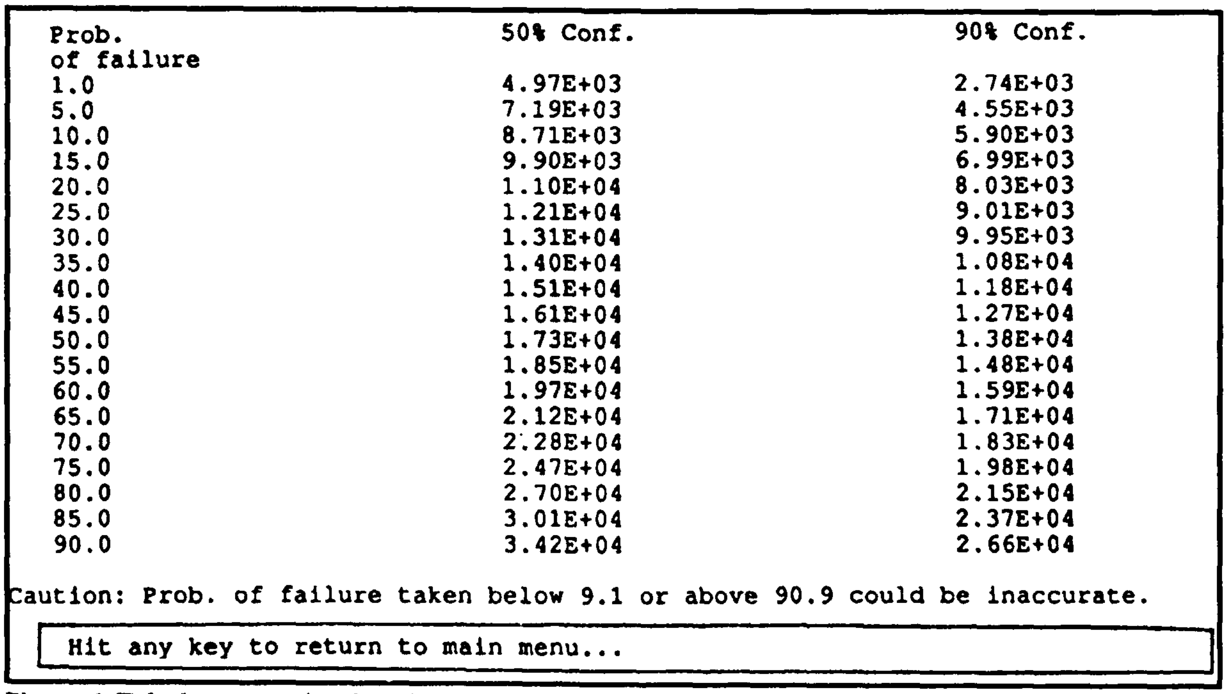
```
Devices        Num      Dose    Num.     Dose    Num.     Dose
Number= 54     1

Dose=
```

Figure 5. Blank table screen (option 1).

is approximated as 66,666,664). After you press the enter key, the number typed will appear in the table and a message at the bottom of the screen will ask whether the last value entered is to be repeated for the next entry. If so, type "Y" and the same value will appear in th . table for the next entry. If not, type "N" and enter the next data point in the manner just described. When the last data point is entered and appears in the table, a message at the bottom of the screen will ask if the data are to be changed. If the data in the table are incorrect, type "Y" and reenter all the data again. If all the data have been entered in the table correctly, type "N" and shortly afterwards the Main Menu will appear on the screen. The raw data have now been entered. By using other options in the Main Menu, you can store the raw data in a file or view the OSLTL values on the PC screen, or print them out in either a tabular or a graphical form.

3.2 Display Results (Option 2)

After raw data have been entered, the OSLTL results can be displayed on the PC screen by typing the number 2 while in the Main Menu screen. A table will appear (see fig. 6), with the left column listing 19 levels (1, 5, 10, 15 ... 90) for the percentage of probability of failure. The OSLTL value associated with a particular probability level is listed in the row to the right of that level. Two values are given: the first value is for 50-percent confidence and the second for 90 percent. When you have finished viewing the information on the screen, press any key on the keyboard. The table will be replaced by the Main Menu screen. If you desire a hard copy of the table see section 3.3.



```
Prob.                        50% Conf.                     90% Conf.
of failure
1.0                          4.97E+03                      2.74E+03
5.0                          7.19E+03                      4.55E+03
10.0                         8.71E+03                      5.90E+03
15.0                         9.90E+03                      6.99E+03
20.0                         1.10E+04                      8.03E+03
25.0                         1.21E+04                      9.01E+03
30.0                         1.31E+04                      9.95E+03
35.0                         1.40E+04                      1.08E+04
40.0                         1.51E+04                      1.18E+04
45.0                         1.61E+04                      1.27E+04
50.0                         1.73E+04                      1.38E+04
55.0                         1.85E+04                      1.48E+04
60.0                         1.97E+04                      1.59E+04
65.0                         2.12E+04                      1.71E+04
70.0                         2.28E+04                      1.83E+04
75.0                         2.47E+04                      1.98E+04
80.0                         2.70E+04                      2.15E+04
85.0                         3.01E+04                      2.37E+04
90.0                         3.42E+04                      2.66E+04

Caution: Prob. of failure taken below 9.1 or above 90.9 could be inaccurate.

 Hit any key to return to main menu...
```

Figure 6. Tabular screen (option 2).

3.3 Print Results (Option 3)

After you have entered the raw data into the computer, you can print out the OSLTL table described in section 3.2 along with the raw data used to generate the table by typing the number 3 while in the Main Menu screen. The message "Printing Results" will appear at the bottom of the PC screen. When printing is complete, the message will change to "Select One", indicating that the program is back to the main menu routine and that other main menu options can now be selected.

Figure 1 is a printout generated by option 3. The table is in the top half of the page while the raw data are at the bottom and are labeled "Fail. Level".

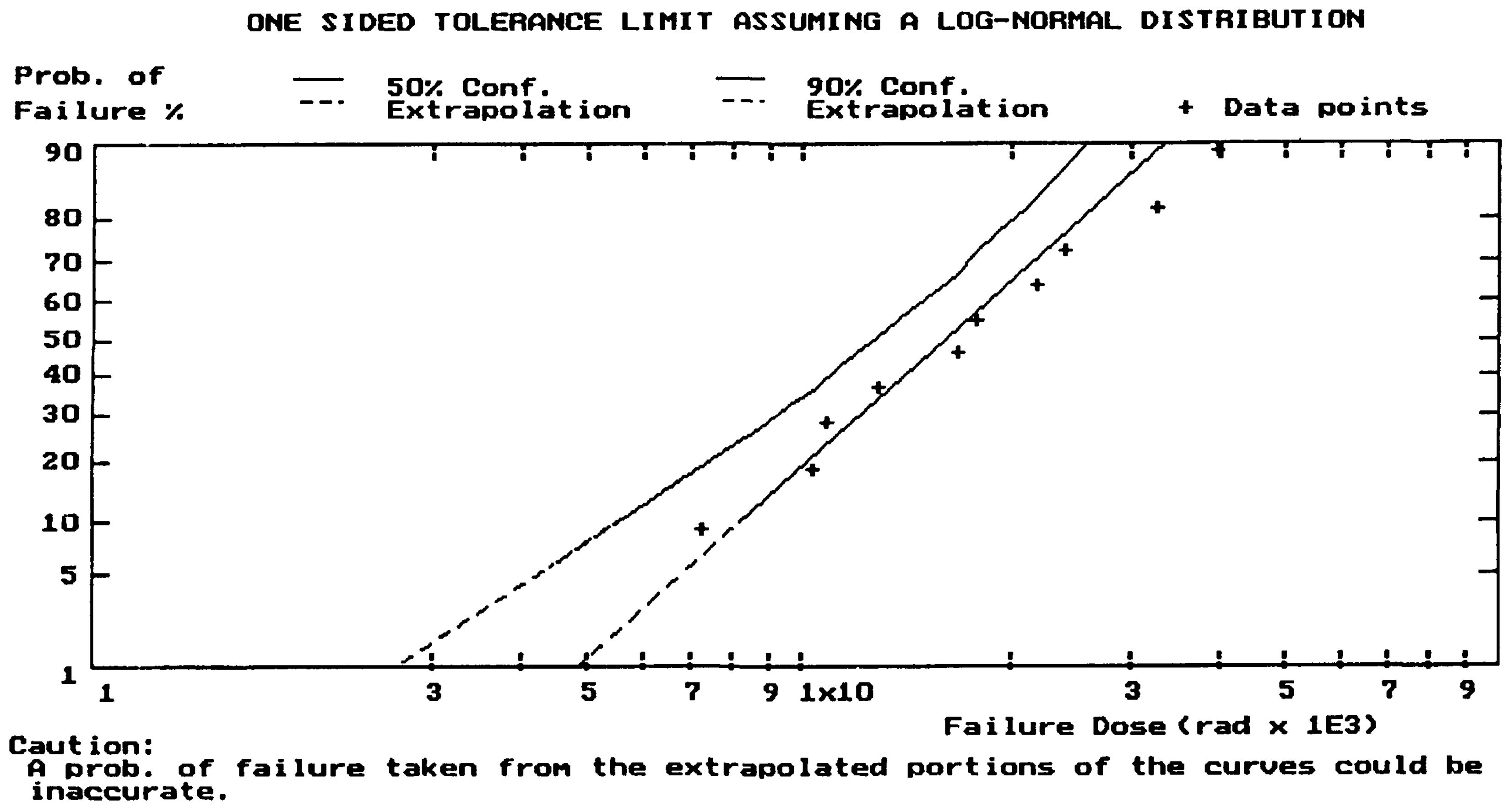
3.4 Display Graph (Option 4)

After raw data have been entered, you can display the OSLTL result in graph form on the PC screen by typing the number 4 while in the Main Menu screen (see fig. 7). The vertical axis is a normal distribution axis (nonlinear axis) representing the percentage of probability of failure; the horizontal is a logarithmic axis representing the failure environment. Two curves are drawn: a solid red curve representing the OSLTL at 90-percent confidence and a solid green straight line representing the same at 50-percent confidence. Both curves are extrapolated using dashed lines. The raw data (individual data points) are plotted using the plus symbol. Section 2 explains the rule used for positioning the data points on the graph. If the raw data have more than nine data points, some data points (plus signs) will be shown above the graph (above the 90-percent probability-of-failure line). After viewing the graph, you can recall the Main Menu screen by pressing any key on the keyboard. If you desire a hard copy of the graph, see section 3.5.

The screen displays from 1 to 3 logarithmic decades on the graph's horizontal axis as required. If a graph with more than 3 decades is required because of the variability of the raw data, the messages "Out of range data" and "Press any key" will appear at the bottom of the screen. No graph will be displayed. Whenever this occurs, the Main Menu remains on the screen but will not accept an input (numbers 1 to 8) until after any key on the keyboard has been pressed. For example, if the number 2 is pressed, the above message disappears and the program is ready for the Main Menu inputs (numbers 1 to 8). If the number 2 is pressed again, the OSLTL results will be displayed.

3.5 Print Graph (Option 5)

After entering the raw data, you can print out the OSLTL graph described in section 3.4, along with the raw data used to generate the



Caution:
A prob. of failure taken from the extrapolated portions of the curves could be inaccurate.

Figure. 7. Graph screen (option 4).

graph, by typing the number 5 while in the Main Menu screen. The Main Menu screen disappears while the message "Printing Graph" appears at the bottom of the screen. When printing is complete, the message disappears while the Main Menu screen, including the message "Select One", reappears.

Figure 2 is a printout generated by option 5. The graph is at the top of the page with the raw data (data points) under the column "Rad. Level" at the bottom of the page. The difference between this graph and the one on the PC screen (option 4) is that the top and left borders of this graph are not drawn, while the legend has been moved from a position above the graph to one below it.

Like the screen display (option 4), the printout uses from 1 to 3 logarithmic decades on the graph's horizontal axis as required. If the variability of the raw data requires a graph with more than 3 decades, the messages "Out of range data" and "Press any key" will appear at the bottom of the screen and no graph will be printed. Whenever this occurs, the Main Menu remains on the screen but will not accept an input (numbers 1 to 8) until after any key has been pressed.

3.6 Save File (Option 6)

You can store the raw data entered in a data file on a hard or a floppy disk by typing the number 6 while in the Main Menu screen. A message at the bottom of the screen indicating "Enter file name" replaces the Main Menu screen. You first type in the drive designation followed by a colon; then you type the new file name and press the enter key. Next the screen message changes to "Saving", indicating that the raw data are being stored. Once storage is complete, the Main Menu screen reappears.

The file is stored by the rules of MS DOS under the file name typed (no more than an eight-character name with no more than a three-character extension, etc.) in the directory specified or working directory when not specified. The file is loaded from disk into the OSTL program through option 7 (see sect. 3.7), and you can erase it from permanent memory through MS DOS (option 8—see sect. 3.8) by using the DEL command.

3.7 Retrieve File (Option 7)

Once raw data have been saved on disk through option 6 (Save File) of the main menu routine, you can load the data into the OSLTL program at a later time by typing the number 7 while in the Main Menu screen. A message at the bottom of the screen indicating "Enter file name" replaces the Main Menu screen. The drive and the file name are then typed and the enter key pressed. Next the screen message changes to "Loading", indicating that the raw data are being retrieved. Once the loading is complete, the Main Menu screen reappears. Afterwards, the user usually exercises either option 2, 3, 4, or 5.

3.8 Exit (Option 8)

The OSLTL program file was loaded and started through MS DOS control. To return the PC to MS DOS control, type the number 8 while in the Main Menu screen. The Main Menu screen will be replaced by the MS DOS prompt.

3.9 Limitations on the Raw Data

The accuracy of the program results depends mainly on the adequacy of the raw data entered. For the total dose, the radiation data are usually generated in one of two ways: in-flux testing or step-stress testing. In in-flux testing, the part is tested during irradiation, resulting in a failure level that is accurate within the tolerances of the test instrumentation, within the accuracy of the radiation failure level determination, and within any annealing of the radiation damage that occurred up to the instant of measurement. However, step-stress testing only results in the failure level being bounded within a range (within an interval). A part is exposed to a certain radiation level and tested afterwards. If failure did not occur, or if sufficient parameter degradation did not occur, the part is incrementally exposed to additional radiation levels until failure occurs. Although most testing is done by the step-stress method, in-flux testing results are preferred for more accurate failure level determination.

The program assumes that the raw data are log-normally distributed as is usually the case for many semiconductor parts. To aid in obtaining log-normal data, the parts should be from the same diffusion lot (especially for sample sizes slightly greater than 10) and be tested under identical test conditions. You can check for log-normality of data generated by in-flux testing by looking at the graph generated by the program (options 4 and 5). The data are roughly log-normally distributed if the plus signs are in an approximate straight line. Because step-stress testing only bounds the failure between a lower and an upper radiation level, it is more difficult to determine whether the data are approximately log-normally distributed, especially for smaller sample sizes. In order for the program to estimate the mean and standard deviation for step-stress data, the radiation steps must be chosen such that failure levels are found in at least four separate intervals (steps). The conservative approach is to use the lowest radiation levels of each interval as the assumed failure level for those parts that failed within that interval. If the number of intervals where failure occurred is greater than 4, or if more than 10 parts were irradiated, the midpoint of the interval can be assumed to be the failure level.

As we explained in section 2, some equations are used to calculate an approximate value for the K_{TL} factor. The larger the sample size (number of parts tested), the closer that approximation is to the actual value of K. At a sample size of 4, the calculated value can be off by a few percent, but at a size of 10 it is off by less than 0.1 percent. Also, the sample size must be large enough to be statistically representative of the lot of parts being tested (i.e., enough parts have to be tested to obtain a good approximation for the mean and standard deviation of the data). Therefore, for accuracy, the sample size should be 10 or greater.

Because of the limits of extrapolation, the accuracy of the results decreases with very large or very small failure probabilities, especially for small sample sizes. Therefore, the graph and table when printed have a caution note indicating that probabilities of failure below $1/n+1$ and above $n/n+1$ are extrapolations and may not be accurate. The statistics, used in calculating the probability of failure, assume that the data were taken from a population of parts with an exact log-normal failure distribution and that the parts irradiated were an accurate statistical representation of that population. In reality, many part types are only approximately log-normally distributed, and with the small number of parts usually tested (in the tens) an accurate statistical representation of the population is not always achieved.

4. Software Description

A copy of the source listing of the OSLTL program is supplied in appendix B. The program is written in the Turbo C language. Page 33 of the program defines some symbols and gives some tables. Pages 34 and 35 contain some routines called "Video functions" used to support the screen graphics. Page 36 contain some routines called "Printer functions" used to support printer graphics. Pages 36 through 39 contain miscellaneous routines. Pages 40 and 41 contain routines referred to as "Math functions" used in the calculation of the OSLTL values.

Pages 41 through 55 contain the seven main routines of the program referred to as option functions, which correspond to the options on the main menu screen:

Option 1	Enter data

Option 2	Display results
Option 3	Print results
Option 4	Display graph
Option 5	Print graph
Option 6	Save file
Option 7	Retrieve file

Program execution starts on page 55 with the "switch" command which sends the program to the "main_menu" routine on page 36. The main menu routine displays the Main Menu screen and waits for a valid keyboard entry (numbers 1 through 8). When you strike a valid key, the program is transferred back to the switch command to select the proper option (cases 1 through 8).

4.1 Enter Data (Option 1)

For option 1, the program transfers to the "enter_data" routine on pages 41 and 42. First, the program displays a screen message (data =) that asks how many raw data points are to be entered and waits for a valid keystroke (numbers 4 through 54). Then the program displays a message (Units =) that asks whether the data are in "rads"or in "n/sq cm" and waits for a valid key stroke (numbers 1 or 2). Afterwards the program asks if you want to change the two selections just made. When the "n" key for no is pressed, the program displays a new screen for entering data and lists the data as they are entered. After all data are entered, the program transfers to the "calculate" routine on page 40, where the OSLTL is calculated for a number of probability-of-failure values. The calculate routine, in conjunction with the "av_ln", "sd_ln", and "K" routines on page 40, uses equations (1) through (5) in section 2 and the "Standard Normal Variable Table" in appendix A (see the "z_p" table on page 33 of the program). After the calculate route is executed, the program returns to the main menu routine through the switch command on page 55.

4.2 Display Results (Option 2)

For option 2, the program transfers to the "case 2" routine on page 56. The case 2 routine immediately transfers the program to the "display_results" routine on page 42, where the table headings are placed on the PC screen and the OSLTL values are calculated using equations (1) through (5) in section 2 and the "Standard Normal Variable Table" in appendix A. Next the results are displayed on the screen and the program is transferred back to the "case 2" routine. The program then waits for any keystroke to return to the "main menu"routine by way of the "switch" command.

4.3 Print Results (Option 3)

For option 3, the program transfers to the "case 3" routine on page 56, where the "Print results" message is placed on the screen. The program is then transferred to the "print_out" routine on page 43, where the labels for the tolerance limit table are printed, the tolerance limit values are calculated using equations (1) through (5) in section 2 in conjunction with the Standard Normal Variable Table in appendix A, and the results are then printed. Next, the routine prints the labels for the raw data table and the raw data. The program then returns to the main menu routine by way of the "case 3" routine and the "switch" command.

4.4 Display Graph (Option 4)

For option 4, the program is transferred to the "case 4" routine on page 56. The case 4 routine immediately tr .nsfers the program to the "fscale" routine on page 37, where the OSLTL values at both 50- and 90-percent confidence are checked to determine whether more than 3 decades are needed for the graph. If so, the messages "Out of range" and "Press any key" are printed on the screen. After the next key stroke, the program returns by way of the case 4 routine and the switch command on page 56 to the "main menu" routine on page 36.

If no more than 3 decades are required, the program goes from the fscale routine by way of the case 4 routine to the "plot" routine on page 45. The plot routine detects the graphic driver (e.g., CGA, EGA, VGA, etc.) and the graphics mode. Based on this information, the horizontal and vertical ratios, cc and rr, respectively, are determined relative to a default screen size (320 × 200 pixels). The scale of the OSLTL data ($\times 10^3$, $\times 10^4$, etc.) is determined. Then a yellow rectangle is drawn to border the graph area that will contain the 50- and 90-percent confidence curves and the plotted raw data points.

The vertical scale is marked at both sides of the graph with the "–" character and the horizontal scale at the top and bottom of the graph with the " | " character.

Then the raw data points are sorted by increasing failure levels using the "q sort" function and are plotted in light-cyan-colored "+" signs. Section 2 describes how the corresponding probability of failure is calculated for each point when plotting raw data on a graph. During the plotting of the data points, the vertical distance from the reference point that corresponds to $n/n{+}1$ is stored in $r2$ and ru, and that for

$1/n+1$ is stored in $r3$ and rl. A solid light-red line representing the 50-percent confidence curve is drawn. Since it is a straight line, only the points at $n/n+1$ and $1/n+1$ percent probability of failure $((c_2, r_2)$ and $(c_3, r_3))$ are used to generate it. The extrapolated lines beyond the data points limit are then drawn. The upper part is drawn in dotted light red between $((c_1, r_1)$ and $(c_2, r_2))$ and the lower part is drawn between $((c_3, r_3)$ and $(c_4, r_4))$.

The 90-percent confidence is drawn as a straight line between the consecutive points on the curve that are stored in the *xl* [*i*] [1] array. Those points that are above the upper limit of the data points (*ru*) are drawn as a dotted light-green curve. Those points that are within the data points (between *ru* and *rl*) are drawn as a solid light-green curve. Finally, the points that are below the lower limit of the data points (*rl*) are drawn as a dotted light-green curve.

To locate points within the graph area, the program uses the upper left corner of the graph—pixel position (17, 30)—as the reference point. This is the graph's minimum failure dose (1×10^x) and the 90-percent probability-of-failure point. From this reference point, the pixel numbers increase when going right along the horizontal axis as well as when going down along the vertical axis. We determined the distance to a pixel by multiplying the horizontal and vertical pixel numbers by the horizontal and vertical ratios (cc and rr), respectively. The horizontal and vertical distances from the reference to another point were determined by the "cl( )" routine on page 37 and the "r[i]" array on page 33.

The "cl( )" routine determines how many log decades are needed by the horizontal axis by using the "max_scale" and "min_scale" values generated in the "f_scale" routine earlier in the program and makes *a* = 1 for one cycle, 2 for two cycles, and 3 for three cycles. Since the horizontal axis is a logarithmic axis, the ratio of the distance in pixels from the reference point to the failure point versus the length of the axis (300 pixels) is equal to the ratio of the log of the difference between the failure point and the reference point values (log *x*) versus the log of the difference between the end of the axis and the reference point values (same as the value of "a"). So

$$\text{distance to failure point} = (300 \log_{10} x)/a \ .$$

The actual pixel position is this distance plus 17 (the horizontal pixel position of the reference point).

The vertical distance from the reference point to the 90-, 85-, 80- ... and 1-percent probability-of-failure points is stored in the "*r*[*i*]" array as *r*[0], *r*[1], *r*[2] ... and *r*[18], respectively. As an example of how these distances are calculated, we will derive the value of *r*[2]—i.e., the distance between 90- (the reference point) and 80-percent probability of failure. From the standard normal variable table in appendix A, Z_p = 1.28 for the 90-percent probability of failure (i.e., 10-percent probability of survival) and Z_p = 2.33 for 1-percent probability of failure. The difference between these numbers is 3.61 and represents the length of the vertical axis. The "rectangle" command on page 46 uses pixel numbers 30 and 150, which make a vertical axis length of 120 pixels. The ratio of the difference between a pixel position and the reference point to the difference between another pixel position and the reference point is equal to the ratio of their Z_p differences. So

$$r[x] = \{(Z_p[x] + 1.28)/3.617\}120.$$

Thus for 80-percent probability of failure $Z_p = -0.84$ and $r[2] = \{(-0.84 + 1.28)/3.61\}120 \approx 15$ pixels.

The actual pixel position is the *r*[*x*] + 30 (vertical pixel position of the reference point).

4.5 Print Graph (Option 5)

For option 5, the program is transferred to the "case 5" routine on page 56. The case 5 routine immediately transfers the program to the "f_scale" routine on page 37, where the OSLTL values at both 50- and 90-percent confidence are checked to determine whether more than 3 decades are needed for the horizontal axis. If so, the messages "out of range" and "press any key" are printed on the screen. After the next key stroke, the program returns by way of the case 5 routine and the switch command on page 56 to the "main_menu" routine on page 36.

If no more than 3 decades are required, the program goes from the "fscale" routine by way of the case 5 routine to the "print_graph" routine on page 50. The print_graph routine initializes the printer by the "dot_pos ("300", "300")" function and the "left_margin" routine. This establishes the reference point of the graph to be in the upper left corner at dot position (300, 300). This is the graph's minimum failure dose (1×10^x) and the 90-percent probability-of-failure point. From this reference point, the dot numbers increase when going right along the horizontal axis as well as when going down along the vertical axis. The horizontal and vertical distances from this reference to another point on the graph were determined by the "pcl( )" routine on page 37 and the "pr[i]" array on page 33. The "pcl()" routine is almost identical to the "cl( )" routine, and the values for the pr[i] array were derived in the same manner as the r[i] array. See section 4.4 for an explanation of how the cl( ) routine and the r[i] array are used to determine the distance from the reference to a particular point on the graph. The position of any point on the graph is then determined by adding 300 (position of the reference point) to the distance from the reference.

The program then determines the scale of the data ($\times 10^3$, $\times 10^4$, etc.). Next, the horizontal and vertical axes are drawn by use of the "." character (dot character). The scale marks are placed on the vertical axis by use of the "–" character. Afterwards, the scale values are placed opposite the scale marks by using the same distance values as are found in the pr[ ] array. The "!" character is used to mark the horizontal scale. The scale is marked for one cycle ana then the scale values are placed beside the marks. The marking and scale value labeling are done a cycle at a time whenever more than one cycle is required.

Next the labels located above the graph are printed. Then the raw data points are sorted by increasing failure levels using the "qsort" function and are plotted using the plus sign. Section 2 describes how the corresponding probability of failure is calculated for each point when plotting raw data on a graph. Afterwards the graph information located below the graph is printed. Finally the 50-percent and 90-percent curves are drawn in almost exactly the same way as in option 4. The solid light-red line representing the 50-percent confidence is replaced by a closely dotted line. The 90-percent confidence curve is drawn as a solid curve. The extrapolated curves are plotted as dotted lines.

On the same page below the graph, the raw data are printed in tabular form. Space is available for up to 3 columns of data at 18 raw data points per column. The program determines how many columns are needed for the data and then prints "DEV #" and "RAD LEVEL" for each column needed. Afterwards, the device number and radiation level are printed.

4.6 Save File (Option 6)

For option 6, the program is transferred to the "case 6" routine on page 56. The case 6 routine immediately transfers the program to the "save ( )" routine on page 54. This routine prompts for a file name from the user by placing a flashing message "Enter file name" in a white rectangle at the bottom of the screen. If the path is not included in the file name, the file will be saved on the current working directory. When a file name is entered, the program opens a communication channel with the disk drive. Then the message "Saving" replaces the "Enter file name" message on the screen. The program then saves the file and returns to the main menu routine by way of the switch command.

4.7 Retrieve File (Option 7)

For option 7, the program is transferred to the "case 7" routine on page 56. The case 7 routine immediately transfers the program to the "load" routine on page 55. This routine prompts for a file name from the user by placing a flashing message, "Enter File Name", in the white rectangle at the bottom of the screen. If the path is not included in the file name, the computer searches the current working directory for the file. When a file name is entered, the program opens a communication channel with the disk drive. Then the message "loading" replaces the "Enter File Name" message on the screen. The program then loads the file and afterwards closes it. The file contains only the raw data. So in order to calculate the OSLTL points for the 50- and 90-percent confidence curves, the program jumps to the "calculate ( )" routine on page 40. Afterwards the program returns to the main menu routine by way of the switch command.

4.8 Exit (Option 8)

For option 8, the program is transferred to the "case 8" routine on page 56. The program then sets the screen to Text mode (mode number 3), exits the program, and returns to the MS DOS prompt.

5. Summary

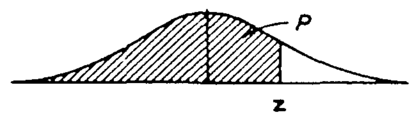
A program has been written in the Turbo C language to calculate the OSLTL for log-normally distributed radiation data. The program will work with numerous IBM or compatible XT, AT, or 386 PCs that use various graphics drivers. However, a printer that uses the PCL format is needed for the hard copies supplied by options 3 and 5.

This report has described the statistics used in the program, has given instructions for the use of the program, and has explained the program's source listing. The program is available for anyone's use and can be modified by them. For a free 5-1/4-in. floppy disk containing the program, contact

Harry Diamond Laboratories
ATTN: SLCHD-NW-TS
2800 Powder Mill Road
Adelphi, MD 20783-1197
(301) 394-3070

Appendix A.—Standard Normal Variable Table

CUMULATIVE NORMAL DISTRIBUTION — VALUES OF z



Values of z_P corresponding to P for the normal curve.

z is the standard normal variable

P	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
.00	—	−2.33	−2.05	−1.88	−1.75	−1.64	−1.55	−1.48	−1.41	−1.34
.10	−1.28	−1.23	−1.18	−1.13	−1.08	−1.04	−0.99	−0.95	−0.92	−0.88
.20	−0.84	−0.81	−0.77	−0.74	−0.71	−0.67	−0.64	−0.61	−0.58	−0.55
.30	−0.52	−0.50	−0.47	−0.44	−0.41	−0.39	−0.36	−0.33	−0.31	−0.28
.40	−0.25	−0.23	−0.20	−0.18	−0.15	−0.13	−0.10	−0.08	−0.05	−0.03
.50	0.00	0.03	0.05	0.08	0.10	0.13	0.15	0.18	0.20	0.23
.60	0.25	0.28	0.31	0.33	0.36	0.39	0.41	0.44	0.47	0.50
.70	0.52	0.55	0.58	0.61	0.64	0.67	0.71	0.74	0.77	0.81
.80	0.84	0.88	0.92	0.95	0.99	1.04	1.08	1.13	1.18	1.23
.90	1.28	1.34	1.41	1.48	1.55	1.64	1.75	1.88	2.05	2.33

Special Values

P	.999	.995	.990	.975	.950	.900
z_P	3.090	2.576	2.326	1.960	1.645	1.282

Appendix B.—Source Listing of Program

```
/********************************************************************************/
/*       This program calculates the failure level of a number of devices      */
/*       and gives the results in a tabular and graphics form, either on       */
/*       the monitor screen or from a laser printer using PCL commands.        */
/*                                                                              */
/*                              Ahmed A. Abou-Auf                               */
/*                                                                              */
/*                          Harry Diamond Laboratories                          */
/*                                                                              */
/*                              TREE/SGEMP Branch                               */
/*                                                                              */
/*                                 18 Jul 91                                    */
/*                                                                              */
/********************************************************************************/

#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
#include<math.h>
#include<conio.h>
#include <dos.h>
#include <string.h>
#include <graphics.h>
#define VEDIO 0x10
#define ESC 27
/*************************************/
/*         Decelerations             */
/*************************************/
FILE *pfile;
FILE *outf;
FILE *inf;
int n;
int un;
float max_scale;
float min_scale;
float pb[19]={10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85,90,95,99};
int r[19]={  0, 8,15,20,25,30,34,38,43,47,51,56,60,65,71,77,85,97,120};
unsigned int
pr[19]={0,70,121,168,209,244,283,316,351,387,420,459,494,535,582,637,703,802,990};
float ln_xl[19][2];
float xl[19][2];
float dose[60];
float zp[10][10]={
{-1000,-2.33,-2.05,-1.88,-1.75,-1.64,-1.55,-1.48,-1.41,-1.34},
{-1.28,-1.23,-1.18,-1.13,-1.08,-1.04,-0.99,-0.95,-0.92,-0.88},
{-0.84,-0.81,-0.77,-0.74,-0.71,-0.67,-0.64,-0.61,-0.58,-0.55},
{-0.52,-0.50,-0.47,-0.44,-0.41,-0.39,-0.36,-0.33,-0.31,-0.28},
{-0.25,-0.23,-0.20,-0.18,-0.15,-0.13,-0.10,-0.08,-0.05,-0.03},
{0.00,0.03,0.05,0.08,0.10,0.13,0.15,0.18,0.20,0.23},
{0.25,0.28,0.31,0.33,0.36,0.39,0.41,0.44,0.47,0.50},
{0.52,0.55,0.58,0.61,0.64,0.67,0.71,0.74,0.77,0.81},
{0.84,0.88,0.92,0.95,0.99,1.04,1.08,1.13,1.18,1.23},
{1.28,1.34,1.41,1.48,1.55,1.64,1.75,1.88,2.05,2.33}};
```

```
/*************************************/
/*         Video Functions          */
/*************************************/

void gotoxy(int x, int y)
{

       union REGS regs;
       regs.h.ah=2;
       regs.h.dh=y;
       regs.h.dl=x;
       regs.h.bh=0;
       int86(VEDIO, &regs, &regs);
}


char attribute(char blink, char back_color,char intensity, char fore_color)
 {
       char c=0;
       blink= blink*128;
       back_color=back_color*16;
       intensity=intensity*8;
       c= blink | back_color | intensity | fore_color;
       return (c);
}

void set(int mode)
{
       union REGS regs;
       regs.h.ah=0;
       regs.h.al=mode;
       int86(VEDIO, &regs, &regs);
}

/*
void str_color(char x, char y, char *str, char blink, char intensity,
       char bcolor, char fcolor)
{
       union REGS regs;
       char l;
       l=strlen(str);      regs.h.ah=9;
       regs.h.bh=0;
       regs.h.bl=attribute(blink, bcolor, intensity, fcolor);
       regs.x.cx=1;
       while(l-!=0)
              {
              regs.h.al=*(str++);
              gotoxy(x++,y);
              int86(VEDIO, &regs, &regs);
              }
}
*/
void str_color(char x, char y, char *str, char blink, char intensity,
       char bcolor, char fcolor)
```

```
{
        textattr(fcolor+(bcolor<<4)+blink*128);
        gotoxy(x,y);
        cputs(str);
}
void write_dot(int color,int col, int row)
{
        union REGS regs;
        regs.h.ah=0x0c;
        regs.h.al=color;
        regs.h.cl=col & 0xff;
        regs.h.ch=(col/256) & 0xff;
        regs.h.dl=row & 0xff;
        regs.h.dh=(row/256) & 0xff;
        int86(VEDIO, &regs, &regs);
}


void scroll_up(char n,char attribute, char u_row, char d_row, char l_col,
        char r_col)
{

        union REGS regs;
        regs.h.ah=0x07;
        regs.h.al=n;
        regs.h.bh=attribute;
        regs.h.cl=l_col;
        regs.h.ch=u_row;
        regs.h.dl=r_col;
        regs.h.dh=d_row;
        int86(VEDIO, &regs, &regs);
}


void cls(char bcolor, char fcolor)
{

        scroll_up(0,attribute(0,bcolor,0,fcolor),0,24,0,79);
}


void  clw(char bcolor, char fcolor, char u_row, char d_row, char l_col,
        char r_col)
{
        scroll_up(0,attribute(0,bcolor,0,fcolor),u_row,d_row,l_col,r_col);
}

void icon(char *s)
{
        /* this function writes a message at the */
        /* bottom line of the screen */

        clw(WHITE,0,24,24,0,79);
        str_color(0,24,s,0,0,WHITE,BLACK);
}
```

```
/*************************************/
/*       Printer Functions           */
/*************************************/

void dot_pos(char *xx, char *yy)
{
        fprintf(pfile,"%c%s%s%c%s%c",ESC,"*p",xx,'x',yy,'Y');
}

void left_margin()
{
        fprintf(pfile,"%c%s",ESC,"*r1A");
}

void end()
{
        fprintf(pfile,"%c%s",ESC,"*rB");
}
void data(char d)
{
        fprintf(pfile,"%c%c%c%c%c%c",ESC,'*','b','1','W',d);
}
void reset()
{
        fprintf(pfile,"%c%c",ESC,'E');
}

void pwrite_dot(unsigned int x, unsigned int y, char *p)
{
        char yy[8], xx[8];
        itoa(x,xx,10);
        itoa(y,yy,10);
        dot_pos(xx,yy);
        fprintf(pfile,"%s",p);
}


/*************************************/
/*       Misc. Functions             */
/*************************************/

char main_menu()
{
        char c=0;
        cls(0,15);
        clw(GREEN, BLACK,5,17,23,55);
        str_color(33,7,"MAIN MENU",0,1,RED,YELLOW);
        str_color(28,9,"1. ENTER DATA",0,1,GREEN,YELLOW);
        str_color(28,10,"2. DISPLAY RESULTS",0,1,GREEN,YELLOW);
        str_color(28,11,"3. PRINT  RESULTS ",0,1,GREEN,YELLOW);
        str_color(28,12,"4. DISPLAY GRAPH",0,1,GREEN,YELLOW);
        str_color(28,13,"5. PRINT GRAPH ",0,1,GREEN,YELLOW);
        str_color(28,14,"6. SAVE FILE",0,1,GREEN,YELLOW);
        str_color(28,15,"7. RETREIVE FILE",0,1,GREEN,YELLOW);
        str_color(28,16,"8. EXIT",0,1,GREEN,YELLOW);
        clw(WHITE,0,24,24,0,79);
        str_color(5,24,"Select one...",0,0,WHITE,BLACK);
        str_color(28,24,"(1,2..., or 8)",1,0,WHITE,RED);
        while( c>57 || c<49 )
        {
                c=getch();
                if(c<=57 && c>=49)
                        return(c);
        }
}

int cl(float x)
{
        int pixel;
        float a,y;
         a=1;
        if ((max_scale - min_scale)==2)
                a=3;
        if ((max_scale - min_scale)==1)
                a=2;
        y=1000*log10(x);
        pixel=y*300/(a*1000);
        return(pixel);
}

int pcl(float x)
{
        int pixel;
        float a,y;
        a=1;
        if ((max_scale - min_scale)==2)
                a=3;
        if ((max_scale - min_scale)==1)
                a=2;
        y=1000*log10(x);
        pixel=y*1700/(a*1000);
        return(pixel);
}
int fscale()
{
        float x0,x1;
        x0=log10(xl[0][0]);
        x1=log10(xl[0][1]);
        if (x0 >= x1)
                max_scale=floor(x0);
        else
                max_scale=floor(x1);
        x0=log10(xl[18][0]);
        x1=log10(xl[18][1]);
        if (x0 <= x1)
                min_scale=floor(x0);
        else
                min_scale=floor(x1);
        if(max_scale - min_scale<=2)
                x0=0;
        else
        {
                printf("\nOut of range data\n");
                printf("Press any key");
                x0=1;
                getch();
        }
        return (x0);
}

int fltcmp(float *f1, float *f2)
{
        int n=0;
        if ( *f1 < *f2)
                n=-1;
        if ( *f1 == *f2)
                n=0;
        if ( *f1 > *f2)
                n=1;
        return(n);

}

char *(input_data)(char str[31])
{
        char *q,c;
        int xi,k,m,i;
        float xf;
        char *p;
        p=malloc(12*sizeof(p));
not_ok:         k=0; m=0;
        strcpy(p,"");
        clw(BLACK,0,22,22,0,79);
        gotoxy(0,22);
        printf("%s",str);
        scanf("%s",p);
        for(i=0; i<strlen(p); i++)
        {
                if(isdigit(p[i])==0)
                {
                        if(p[i]=='E' || p[i]=='e' )
                        {
                                k++;
                                if (k==1 && i>0)
                                {
                                        m=0;
                                        goto ok;
                                }
                                else
                                        goto not_ok;
                        }
                        if (p[i]=='.')
                        {
                                m++;
                                if (m==1)
                                        goto ok;
                        }
                        goto not_ok;
ok:                     ;
                }
        }
        return (p);
}

float get_dose( int j)
{
        int i,x,y;
        float jj;
        char temp[15];
        char c, p[12], q[12];
        jj=j;
        x=(jj-1)/18;
        y=fmod(j-1,18);
        itoa(j,q,10);
        str_color(13+x*21,y+3,q,0,0,CYAN,BLACK);
        str_color(21+x*21,y+3,"          ",0,0,CYAN,BLACK);
        if(j==1)
                goto a;
        clw(WHITE,0,24,24,0,79);
        str_color(5,24,"Do you want to repeat the above value ?
",0,0,WHITE,BLACK);
        str_color(48,24,"( y/n )         ",1,0,WHITE,RED);
        c=0;
        while(c!='y' && c!='n')
                c=getch();
        if(c=='y')
        {
                strcpy(p,temp);
                clw(WHITE,0,24,24,0,79);
        }

        else
a:      {
                clw(WHITE,0,24,24,0,79);
                strcpy(p,input_data("Dose= "));
                strcpy(temp,p);
        }
        str_color(24+x*21,y+3,p,0,0,CYAN,BLACK);
        clw(WHITE,0,24,24,0,79);
        return(atof(p));
}
```

Appendix B

```
/***************************************/
/*           Math Functions            */
/***************************************/

float z( float q )
{
       float x;
       int row, col, i;
       i=ceil(q);
       row= i/10;
       col=i%10;
       x=zp[row][col];
       return(x);
}

float k( float p, float cc)
{
       float a,b,x,y,t;
       t=z(cc)*z(cc);
       y=z(p)*z(p);
       a=1- (t/(2*(n-1)));
       b= y-t/n;
       x=(z(p)+sqrt(y-a*b))/a;
       return(x);
}

float av_ln()
{
       int i;
       float sum=0;
       for(i=0; i<n; i++)
              sum+= log(dose[i]);
       return(sum/n);
}

float sd_ln()
{
       int i;
       float av;
       float sum=0;
       av=av_ln();
       for(i=0; i<n; i++)
              sum+=(log(dose[i]) - av)*(log(dose[i]) - av);
       return(sqrt(sum/(n-1)));
}
void calculate()
{
       float a,b,av,sd;
       int i;
       av=av_ln();
       sd=sd_ln();
       for (i=18; i>=0; i-)
       {
              ln_xl[i][0]=av - k(pb[i],50)*sd;
```

```
            xl[i][0]=exp(ln_xl[i][0]);
            ln_xl[i][1]=av - k(pb[i],90)*sd;
            xl[i][1]=exp(ln_xl[i][1]);
        }
}



/*************************************/
/*        Option 1 Function          */
/*************************************/

void enter_data()
{
        char x,c,p[12];
        int i;
        cls(0,15);
        clw(CYAN, BLACK,0,6,0,9);
a:      clw(WHITE,0,24,24,0,79);
        str_color(5,24,"Enter data...",0,0,WHITE,BLACK);
        str_color(1,1,"Devices",0,1,RED,YELLOW);
        str_color(0,3,"Number=   ",0,0,CYAN,BLACK);
        str_color(0,5,"Units=   ",0,0,CYAN,BLACK);
        str_color(28,24,"(4,5..., or 54)",1,0,WHITE,RED);
        strcpy(p,input_data("Number= "));
        n=atoi(p);
        if(n<=3 || n>54)
                goto a;
        str_color(8,3,p,0,0,CYAN,BLACK);

b:      str_color(28,24,"(1) rads or (2) n/sq cm  (1 or 2) ",1,0,WHITE,RED);
        strcpy(p,input_data("Units= "));
        un=atoi(p);
        if(un!= 1 && un!= 2)
                goto b;
        str_color(8,5,p,0,0,CYAN,BLACK);
        clw(WHITE,0,24,24,0,79);
        str_color(5,24,"Do you want to change your data ?       ",0,0,WHITE,BLACK);
        str_color(48,24,"( y/n )        ",1,0,WHITE,RED);
        c=0;
        while(c!='y' && c!='n')
                c=getch();
        if(c=='y')
                goto a;
        clw(WHITE,0,24,24,0,79);
        clw(CYAN, BLACK,0,20,11,30);
        str_color(12,1,"Num.      Dose   " ,0,1,RED,YELLOW);
        clw(CYAN, BLACK,0,20,32,51);
        str_color(34,1,"Num.      Dose   " ,0,1,RED,YELLOW);
        clw(CYAN, BLACK,0,20,53,72);
        str_color(55,1,"Num.      Dose   " ,0,1,RED,YELLOW);
c: str_color(28,24,"                   ",1,0,WHITE,RED);
        str_color(5,24,"Enter data...",0,0,WHITE,BLACK);
        for(i=1; i<=n; i++)
```

```
            dose[i-1]=get_dose(i);
        clw(WHITE,BLACK,24,24,0,79);
        str_color(5,24,"Do you want to change your data ?         ",0,0,WHITE,BLACK);
        str_color(48,24,"( y/n )          ",1,0,WHITE,RED);
        c=0;
        while(c!='y' && c!='n')
            c=getch();
        if(c=='y')
            goto c;
        calculate();

}
/*************************************/
/*        Option 2 Function          */
/*************************************/

void display_results()
{
        float a,b,av,sd;
        int i;
        av=av_ln();
        sd=sd_ln();
        cls(0,15);
        gotoxy(3,0);
        printf("Prob.");
        gotoxy(33,0);
        printf("50% Conf.");
        gotoxy(63,0);
        printf("90% Conf.");
        gotoxy(3,1);
        printf("of failure");
        for (i=18; i>=0; i-)
        {
            ln_xl[i][0]=av - k(pb[i],50)*sd;
            xl[i][0]=exp(ln_xl[i][0]);
            ln_xl[i][1]=av - k(pb[i],90)*sd;
            xl[i][1]=exp(ln_xl[i][1]);
            gotoxy(3,20-i);
            printf("%.1f",100-pb[i]);
            gotoxy(33,20-i);
            printf("%.3E",xl[i][0]);
            gotoxy(63,20-i);
            printf("%.3E",xl[i][1]);
        }
        a=n+1;
        b=100*n/a;
        a=100/a;
        printf("\n\nCaution: Prob. of failure taken below %.1f or above %.1f could be
inaccurate.",a,b);
}
```

```
/*************************************/
/*        Option 3 Function          */
/*************************************/

void print_out()
{
        float a,b,av,sd,x;
        char p[10];
        char q[10];
        int i,j,nn,mm;
        pfile=fopen("PRN","w");
        dot_pos("300","200");
        left_margin();
        av=av_ln();
        sd=sd_ln();
        fprintf(pfile,"One Sided Tolerance Limit Assuming a Log Normal Distribu-
tion");
        dot_pos("300","350");
        fprintf(pfile,"Prob.");
        dot_pos("800","350");
        fprintf(pfile,"50% Conf.");
        dot_pos("1400","350");
        fprintf(pfile,"90% Conf.");
        dot_pos("300","400");
        fprintf(pfile,"of failure");
        for (i=18; i>=0; i-)
        {
                ln_xl[i][0]=av - k(pb[i],50)*sd;
                xl[i][0]=exp(ln_xl[i][0]);
                ln_xl[i][1]=av - k(pb[i],90)*sd;
                xl[i][1]=exp(ln_xl[i][1]);
                itoa(450+(18-i)*50,p,10);
                dot_pos("300",p);
                fprintf(pfile,"%.1f",100-pb[i]);
                dot_pos("800",p);
                fprintf(pfile,"%.3E",xl[i][0]);
                dot_pos("1400",p);
                fprintf(pfile,"%.3E",xl[i][1]);

        }
        a=n+1;
        b=100*n/a;
        a=100/a;
        fprintf(pfile,"\n\n\tCaution:");
        fprintf(pfile,"\n\t  Prob. of failure taken below %.1f or above %.1f could be
inaccurate.",a,b);
        nn=0; mm=0;
        if(n>=19)
                nn=1;
```

```
if(n>=37)
        mm=2;
itoa(300 ,q,10);
dot_pos(q,"1600");
fprintf(pfile,"Dev.");
dot_pos(q,"1650");
fprintf(pfile,"No.");
itoa(300+ 600*nn ,q,10);
dot_pos(q,"1600");
fprintf(pfile,"Dev.");
dot_pos(q,"1650");
fprintf(pfile,"No.");
itoa(300+ 600*mm ,q,10);
dot_pos(q,"1600");
fprintf(pfile,"Dev.");
dot_pos(q,"1650");
fprintf(pfile,"No.");
itoa(500 ,q,10);
dot_pos(q,"1600");
fprintf(pfile,"Fail. Level");
dot_pos(q,"1650");
if (un==1)
        fprintf(pfile,"rads (Si)");
if (un==2)
        fprintf(pfile,"n/sq cm");
itoa(500+ 600*nn ,q,10);
dot_pos(q,"1600");
fprintf(pfile,"Fail. Level");
dot_pos(q,"1650");
if (un==1)
        fprintf(pfile,"rads (Si)");
if (un==2)
        fprintf(pfile,"n/sq cm");
itoa(500+ 600*mm ,q,10);
dot_pos(q,"1600");
fprintf(pfile,"Fail. Level");
dot_pos(q,"1650");
if (un==1)
        fprintf(pfile,"rads (Si)");
if (un==2)
        fprintf(pfile,"n/sq cm");
for (j=0; j<n; j++)
{
        x=j;
        nn=x/18;
        itoa(300+ 600*nn ,q,10);
        itoa(fmod(j,18)*50 + 1700,p,10);
        dot_pos(q,p);
        fprintf(pfile,"%d",j+1);
        itoa(500+ 600*nn ,q,10);
        dot_pos(q,p);
        fprintf(pfile,"%.3E",dose[j]);
}
end();
```

```
        fprintf(pfile,"\f");
        fclose(pfile);
}

/************************************/
/*          Option 4 Function       */
/************************************/
void plot()
{
        int i, col, row;
        int g_driver, g_mode;
        char p[20], q[5];
        float x,c1,c2,c3,c4,r1,r2,r3,r4,ru,rl,range,rr,cc;
        detectgraph(&g_driver,& g_mode);
        initgraph(&g_driver,& g_mode,"");
        cc=1;  rr=1;
        switch(g_driver)
        {
                case CGA:
                        if(g_mode==4)
                                cc=2;
                        break;
                case MCGA:
                        if(g_mode>=4)
                                cc=2;
                        if(g_mode==5)
                                rr=2.4;
                        break;
                case EGA:
                        cc=2;
                        if(g_mode==1)
                                rr=1.75;
                         break;
                case EGA64:
                        cc=2;
                        if(g_mode==1)
                                rr=1.75;
                         break;
                case EGAMONO:
                        cc=2;
                        rr=1.75;
                         break;
                case HERCMONO:
                        cc=2.25;
                        rr=1.74;
                         break;
                case ATT400:
                        if(g_mode>=4)
                                cc=2;
                        if(g_mode==5)
                                rr=2;
                        break;
                case VGA:
                        cc=2;
                        if(g_mode==1)
                                rr=1.75;
                        if(g_mode==2)
                                rr=2.4;
                        break;

                case PC3270:
                        cc=2.25;
                        rr=1.75;
                        break;

                case IBM8514:
                        if(g_mode==0)
                        {
                                cc=2;  rr=2.4;
                        }
                        if(g_mode==1)
                        {
                                cc=3.2;         rr=3.84;
                        }
                }

        range=pow10(min_scale);
        setcolor(YELLOW);
        rectangle(17*cc,30*rr,319*cc,150*rr);
        for(i=2; i<17; i=i+2)
        {
                outtextxy(17*cc,(28+r[i])*rr,"_" );
                outtextxy(315*cc,(28+r[i])*rr,"_");
        }
        for(i=17; i<18; i++)
        {
                outtextxy(17*cc,(28+r[i])*rr,"_" );
                outtextxy(315*cc,(28+r[i])*rr,"_");
        }
        for(i=3; i<10; i=i+1)
        {
                outtextxy((17+cl(i))*cc,30*rr,"|");
                outtextxy((17+cl(i))*cc,147*rr,"|");
        }
        if((max_scale - min_scale)>=1)
        {
                for(i=10; i<100; i=i+10)
                {
                        outtextxy((17+cl(i))*cc,30*rr,"|");
                        outtextxy((17+cl(i))*cc,147*rr,"|");
                }
        }
        if((max_scale- min_scale)==2)
        {
                for(i=100; i<1000; i=i+100)
                {
                        outtextxy((17+cl(i))*cc,30*rr,"|");
                        outtextxy((17+cl(i))*cc,147*rr,"|");
                }
        }
```

```
setcolor(LIGHTCYAN);
qsort(&dose,n,sizeof(dose[0]),fltcmp);
for (i=0 ; i<n; i++)
{
        col=17+cl(dose[i]/range);
        x=i;
        x=(x+1)/(n+1);
        x=z((1-x)*100);
        x=(x+1.28)*120/3.61;
        row=30+x;
        if(i==n-1)
        {
                r2=row;
                ru=row;
        }
        if(i==0)
        {
                r3=row;
                rl=row;
        }
        outtextxy(col*cc,row*rr,"+");
}
cl=17+cl(xl[0][0]/range);
c4=17+cl(xl[18][0]/range);
rl=30 + r[0];
r4=30 + r[18];
c2=cl + (c4 - cl)*(r2 -rl)/(r4 - rl);
c3=cl + (c4 - cl)*(r3 -rl)/(r4 - rl);
setcolor(LIGHTRED);
setlinestyle(0,0,1);
line(c2*cc,r2*rr,c3*cc,r3*rr);
line(60*cc,15*rr,70*cc,15*rr);
setcolor(LIGHTRED);
setlinestyle(3,0,1);
line(cl*cc,rl*rr,c2*cc,r2*rr);
line(c3*cc,r3*rr,c4*cc,r4*rr);
line(60*cc,20*rr,70*cc,20*rr);
for(i=0; i<18; i++)
{
        cl=17+cl(xl[i][1]/range);
        c4=17+cl(xl[i+1][1]/range);
        r4=30+r[i+1];
        rl=30+r[i];
        if ( rl<ru && r4<=ru )
        {
                setcolor(LIGHTGREEN);
                setlinestyle(3,0,1);
                line(cl*cc,rl*rr,c4*cc,r4*rr);
        }
        if ( rl<=ru && r4>ru )
        {
                r2=ru;
                c2=cl + (c4 - cl)*(r2 -rl)/(r4 - rl);
                setcolor(LIGHTGREEN);
                setlinestyle(3,0,1);
                line(c1*cc,r1*rr,c2*cc,r2*rr);
                setcolor(LIGHTGREEN);
                setlinestyle(0,0,1);
                line(c2*cc,r2*rr,c4*cc,r4*rr);
        }
        if ( r1>=ru && r4<=rl )
        {
                setcolor(LIGHTGREEN);
                setlinestyle(0,0,1);
                line(c1*cc,r1*rr,c4*cc,r4*rr);
        }
        if ( r1<rl && r4>=rl )
        {
                r3=rl;
                c3=c1 + (c4 - c1)*(r3 -r1)/(r4 - r1);
                setcolor(LIGHTGREEN);
                setlinestyle(0,0,1);
                line(c1*cc,r1*rr,c3*cc,r3*rr);
                setcolor(LIGHTGREEN);
                setlinestyle(3,0,1);
                line(c3*cc,r3*rr,c4*cc,r4*rr);
        }
        if ( r1>=ru && r4>rl )
        {
                setcolor(LIGHTGREEN);
                setlinestyle(3,0,1);
                line(c1*cc,r1*rr,c4*cc,r4*rr);
        }
}
setcolor(LIGHTGREEN);
setlinestyle(0,0,1);
line(150*cc,15*rr,160*cc,15*rr);
setlinestyle(3,0,1);
line(150*cc,20*rr,160*cc,20*rr);
setlinestyle(0,0,1);
setcolor(LIGHTCYAN);
outtextxy(250*cc,20*rr,"+");
setcolor(LIGHTGRAY);
i=-1;
outtextxy((19+cl(i=i+2))*cc,154*rr,"1");
outtextxy((18+cl(i=i+2))*cc,154*rr,"3");
outtextxy((18+cl(i=i+2))*cc,154*rr,"5");
outtextxy((18+cl(i=i+2))*cc,154*rr,"7");
outtextxy((18+cl(i=i+2))*cc,154*rr,"9");
if((max_scale- min_scale)>=1)
{
        i=-10;
        outtextxy((19+cl(i=i+20))*cc,154*rr,"1x10");
        outtextxy((18+cl(i=i+20))*cc,154*rr,"3");
        outtextxy((18+cl(i=i+20))*cc,154*rr,"5");
        outtextxy((18+cl(i=i+20))*cc,154*rr,"7");
        outtextxy((18+cl(i=i+20))*cc,154*rr,"9");
}
```

```
        if((max_scale- min_scale)==2)
        {
              i=-100;
              outtextxy((19+cl(i=i+200))*cc,180*rr,"1x100");
              outtextxy((18+cl(i=i+200))*cc,180*rr,"3");
              outtextxy((18+cl(i=i+200))*cc,180*rr,"5");
              outtextxy((18+cl(i=i+200))*cc,180*rr,"7");
              outtextxy((18+cl(i=i+200))*cc,180*rr,"9");
        }
        i=0;
        outtextxy(7*cc,(30+r[i])*rr,"90");
        outtextxy(7*cc,(30+r[i=i+2])*rr,"80");
        outtextxy(7*cc,(30+r[i=i+2])*rr,"70");
        outtextxy(7*cc,(30+r[i=i+2])*rr,"60");
        outtextxy(7*cc,(30+r[i=i+2])*rr,"50");
        outtextxy(7*cc,(30+r[i=i+2])*rr,"40");
        outtextxy(7*cc,(30+r[i=i+2])*rr,"30");
        outtextxy(7*cc,(30+r[i=i+2])*rr,"20");
        outtextxy(7*cc,(30+r[i=i+2])*rr,"10");
        outtextxy(7*cc,(30+r[i=i+1])*rr," 5");
        outtextxy(7*cc,(30+r[i=i+1])*rr," 1");
        outtextxy(80*cc,15*rr,"50% Conf.");
        outtextxy(170*cc,15*rr,"90% Conf.");
        outtextxy(260*cc,20*rr,"Data points");
        outtextxy(170*cc,20*rr,"Extrapolation");
        outtextxy(80*cc,20*rr,"Extrapolation");
        setcolor(WHITE);
        outtextxy(50*cc,0*rr,"ONE SIDED TOLERANCE LIMIT ASSUMING A LOG-NORMAL DISTRI-
BUTION");
        outtextxy(1,12*rr,"Prob. of");
        outtextxy(1,20*rr,"Failure %");
        outtextxy(200*cc,162*rr,"Failure Dose");
        i=min_scale;
        itoa(i,q,10);
        if (un==1)
              strcpy(p,"(rad x 1E");
        if (un==2)
              strcpy(p,"(n/sq cm x 1E");
        strcat(p,q);
        strcat(p,")");
        outtextxy(250*cc,162*rr,p);
        setcolor(YELLOW);
        outtextxy(0*cc,166*rr,"Caution:");
        setcolor(LIGHTGRAY);
        outtextxy(0*cc,171*rr," A prob. of failure taken from the extrapolated por-
tions of the curves could be");
        outtextxy(0*cc,176*rr," inaccurate.");

}
```

```
/*************************************/
/*        Option 5 Function          */
/*************************************/

print_graph()
{
        int i, j, col, row,nn, mm;
        float c1,c2,c3,c4,r1,r2,r3,r4,ru,rl,x,range;
        char s[4];
        char p[10], q[10];

        pfile=fopen("PRN","w");
        dot_pos("300","300");
        left_margin();
        range=pow10(min_scale);
        for (col=300; col<=2000; col++)
                pwrite_dot(col,1290,".");
        for (row=300; row<=1290; row++)
                pwrite_dot(300,row,".");
        for(i=0; i<17; i=i+2)
                pwrite_dot(305,290+pr[i],"_");
        for(i=17; i<21; i=i+1)
                pwrite_dot(305,290+pr[i],"_");
        pwrite_dot(250,300+0,"90");
        pwrite_dot(250,300+121,"80");
        pwrite_dot(250,300+209,"70");
        pwrite_dot(250,300+283,"60");
        pwrite_dot(250,300+351,"50");
        pwrite_dot(250,300+420,"40");
        pwrite_dot(250,300+494,"30");
        pwrite_dot(250,300+582,"20");
        pwrite_dot(250,300+703,"10");
        pwrite_dot(250,300+791," 5");
        pwrite_dot(250,300+990," 1");
        for(i=1; i<=10; i=i+1)
                pwrite_dot(300+pcl(i),1290,"!");
        for(i=1; i<=10; i=i+2)
        {
                itoa(i,s,10);
                pwrite_dot(300+pcl(i),1330,s);
        }
        if((max_scale - min_scale)>=1)
        {
                for(i=10; i<100; i=i+10)
                        pwrite_dot(300+pcl(i),1290,"!");

                for(i=10; i<=100; i=i+20)
                {
                        x=i;
                        x=x/10;
                        itoa(x,s,10);
                        pwrite_dot(300+pcl(i),1330,s);
                }
                pwrite_dot(315+pcl(10),1330,"x10");
        }
```

```
if((max_scale - min_scale)==2)
{
        for(i=100; i<=1000; i=i+100)
                pwrite_dot(300+pcl(i),1290,"!");
        fcr(i=100; i<=1000; i=i+200)
        {
                x=i;
                x=x/100;
                itoa(x,s,10);
                pwrite_dot(300+pcl(i),1330,s);
        }
        pwrite_dot(315+pcl(100),1330,"x100");
}
qsort(&dose,n,sizeof(dose[0]),fltcmp);
for (i=0 ; i<n; i++)
{
        col=300+pcl(dose[i]/range);
        x=i;
        x=(x+1)/(n+1);
        x=z((1-x)*100);
        x=(x+1.28)*990/3.61;
        row=295+x;
        if(i==n-1)
        {
                r2=row;
                ru=row;
        }
        if(i==0)
        {
                r3=row;
                rl=row;
        }
        pwrite_dot(col,row,"+");
}

c1=300+pcl(xl[0][0]/range);
c4=300+pcl(xl[18][0]/range);
r4=300+pr[18];
r1=300+pr[0];
c2=c1 + (c4 - c1)*(r2 - r1)/(r4 - r1);
c3=c1 + (c4 - c1)*(r3 - r1)/(r4 - r1);
for(row=r2; row<=r3; row=row+7)
{
        col= c2 + (row-r2)*(c3-c2)/(r3-r2);
        pwrite_dot(col,row,".");
}
for(row=r1; row<=r2; row=row+15)
{
        col= c1 + (row-r1)*(c2-c1)/(r2-r1);
        pwrite_dot(col,row,".");
}
for(row=r3; row<=r4; row=row+15)
{
        col= c3 + (row-r3)*(c4-c3)/(r4-r3);
```

```
                pwrite_dot(col,row,".");
        }
        for(i=0; i<18; i++)
        {
                c1=300+pc1(x1[i][1]/range);
                c4=300+pc1(x1[i+1][1]/range);
                r4=300+pr[i+1];
                r1=300+pr[i];
                if ( r1<ru && r4<=ru )
                {
                        for(row=r1; row<=r4; row=row+10)
                        {
                                col= c1 + (row-r1)*(c4-c1)/(r4-r1);
                                pwrite_dot(col,row,".");
                        }
                }
                if ( r1<=ru && r4>ru )
                {
                        r2=ru;
                        c2=c1 + (c4 - c1)*(r2 -r1)/(r4 - r1);
                        for(row=r1; row<=r2; row=row+10)
                        {
                                col= c1 + (row-r1)*(c2-c1)/(r2-r1);
                                pwrite_dot(col,row,".");
                        }
                        for(row=r2; row<=r4; row++)
                        {
                                col= c2 + (row-r2)*(c4-c2)/(r4-r2);
                                pwrite_dot(col,row,".");
                        }
                }
                if ( r1>=ru && r4<=rl )
                {
                        for(row=r1; row<=r4; row++)
                        {
                                col= c1 + (row-r1)*(c4-c1)/(r4-r1);
                                pwrite_dot(col,row,".");
                        }
                }
                if ( r1<rl && r4>=rl )
                {
                        r3=rl;
                        c3=c1 + (c4 - c1)*(r3 -r1)/(r4 - r1);
                        for(row=r1; row<=r3; row++)
                        {
                                col= c1 + (row-r1)*(c3-c1)/(r3-r1);
                                pwrite_dot(col,row,".");
                        }
                        for(row=r1; row<=r3; row=row+10)
                        {
                                col= c3 + (row-r3)*(c4-c3)/(r4-r3);
                                pwrite_dot(col,row,".");
                        }
                }
```

```
          if ( rl>=ru && r4>rl )
          {
                for(row=rl; row<=r4; row=row+10)
                {
                      col= cl + (row-rl)*(c4-cl)/(r4-rl);
                      pwrite_dot(col,row,".");
                }
          }
    }
    for(i=0; i<10; i=i+2)
    {
          write_dot(1,10+i,4);
          write_dot(2,100+i,4);
    }

    pwrite_dot(600,150,"ONE SIDED TOLERANCE LIMIT ASSUMING A LOG-NORMAL DISTRIBU-
TION");
    pwrite_dot(200,200,"Probability");
    pwrite_dot(200,250,"of Failure%");
    pwrite_dot(1700,1390,"Failure Level");
    pwrite_dot(500,1440,"Legend :");
    for (i=0; i<=50; i=i+10)
          pwrite_dot(550+i,1490,".");
    pwrite_dot(650,1490,"50%  Confidence");
    for (i=0; i<=50; i++)
          pwrite_dot(550+i,1540,".");
    pwrite_dot(650,1540,"90% Confidence");
    for (i=0; i<=50; i=i+15)
          pwrite_dot(550+i,1590,".");
    pwrite_dot(650,1590,"Extrapolation");
    pwrite_dot(550,1650,"+");
    pwrite_dot(650,1640,"Data Points");
    pwrite_dot(300,1700,"Caution");
    pwrite_dot(300,1750,"  A probability of failure taken from the extrapolated
portion of the ");
    pwrite_dot(300,1800,"  curves could be inaccurate.");
    dot_pos("1700","1440");
    if (un==1)
          fprintf(pfile,"(rad x 1E%.0f )",min_scale);
    if (un==2)
          fprintf(pfile,"(n/sq cm x 1E%.0f )",min_scale);
    dot_pos("2000","2000");
    nn=0; mm=0;
    if(n>=19)
          nn=1;
    if(n>=37)
          mm=2;
    itoa(300 ,q,10);
    dot_pos(q,"1900");
    fprintf(pfile,"Dev.");
    dot_pos(q,"1950");
    fprintf(pfile,"No.");
    itoa(300+ 600*nn ,q,10);
    dot_pos(q,"1900");
```

```
        fprintf(pfile,"Dev.");
        dot_pos(q,"1950");
        fprintf(pfile,"No.");
        itoa(300+ 600*mm ,q,10);
        dot_pos(q,"1900");
        fprintf(pfile,"Dev.");
        dot_pos(q,"1950");
        fprintf(pfile,"No.");
        itoa(500 ,q,10);
        dot_pos(q,"1900");
        fprintf(pfile,"Fail. Level");
        dot_pos(q,"1950");
        if (un==1)
                fprintf(pfile,"rads (Si)");
        if (un==2)
                fprintf(pfile,"n/sq cm");
        itoa(500+ 600*nn ,q,10);
        dot_pos(q,"1900");
        fprintf(pfile,"Fail. Level");
        dot_pos(q,"1950");
        if (un==1)
                fprintf(pfile,"rads (Si)");
        if (un==2)
                fprintf(pfile,"n/sq cm");
        itoa(500+ 600*mm ,q,10);
        dot_pos(q,"1900");
        fprintf(pfile,"Fail. Level");
        dot_pos(q,"1950");
        if (un==1)
                fprintf(pfile,"rads (Si)");
        if (un==2)
                fprintf(pfile,"n/sq cm");
        for (j=0; j<n; j++)
        {
                x=j;
                nn=x/18;
                itoa(300+ 600*nn ,q,10);
                itoa(fmod(j,18)*50 + 2000,p,10);
                dot_pos(q,p);
                fprintf(pfile,"%d",j+1);
                itoa(500+ 600*nn ,q,10);
                dot_pos(q,p);
                fprintf(pfile,"%.3E",dose[j]);
        }
        end();
        fprintf(pfile,"\f");
        fclose(pfile);
}

/**************************************/
/*      Option 6 Function             */
/**************************************/

void save()
```

```
{
        int i;
        char wf[20];
        cls(0,15);
        icon("Enter file name");
        gotoxy(0,22);
        scanf("%s",wf);
        outf=fopen(wf,"w");
        if (outf==NULL)
        {
                icon("Cannot open specified file - hit any key...");
                while(getch()=='');
        }
        icon("Saving...");
        fprintf(outf,"%d\n",n);
        fprintf(outf,"%d\n",un);
        for(i=0; i<n; i++)
                fprintf(outf,"%f\n",dose[i]);
        fclose(outf);
}

/*************************************/
/*      Option 7 Function           */
/*************************************/

void load()
{
        int i;
        char wf[20];
        cls(0,15);
        icon("Enter file name");
        gotoxy(0,22);
        scanf("%s",wf);
        inf=fopen(wf,"r");
        if (inf==NULL)
        {
                icon("Cannot open specified file - hit any key...");
                while(getch()=='');
        }
        icon("Loading...");
        fscanf(inf,"%d\n",&n);
        fscanf(inf,"%d\n",&un);
        for(i=0; i<n; i++)
                fscanf(inf,"%f\n",&dose[i]);
        fclose(outf);
        calculate();
}


main()

{

a:      switch(main_menu())
```

```
      {
            case '1':
                        enter_data();
                        goto a;
            case '2':
                        display_results();
                        icon("    Hit any key to return to main menu...");
                        while(getch()=='');
                        goto a;
            case '3':
                        cls(0,15);
                        icon("Printing results ...");
                        print_out();
                        goto a;
            case '4':
                        if (fscale()==1)
                        goto a;
                        plot();
                        while(getch()=='');
                        set(3);
                        goto a;
            case '5':
                        if (fscale()==1)
                        goto a;
                        cls(0,15);
                        icon("Printing graph ...");
                        print_graph();
                        goto a;
            case '6':   save();
                        goto a;
            case '7':   load();
                        goto a;
            case '8':   set(3);
      }

}
```

Distribution

Administrator
Defense Technical Information Center
Attn DTIC-DDA (2 copies)
Cameron Station, Building 5
Alexandria, VA 22304-6145

Director
Defense Intelligence Agency
Attn DB-4C, D. Spohn
Attn Technical Library
Washington, DC 20301

Defense Nuclear Agency
Attn RAEE, LCDR L. Cohn
Washington, DC 20305

Director
Defense Research & Engineering
Attn D. Patterson
Washington, DC 20301

Dept CH Staff for Chief of Res, Dev,
& Acquisition
Department of the Army
Attn DAMA-WSA, LTC S. G. Gebert
Washington, DC 20310

Commander
US Army Communication R&D Command
Attn Technical Library
Ft. Monmouth, NJ 07703

Director
US Army Laboratory Command
Attn AMSLC-VL-NE
Vulnerability/Lethality Assessment
Management Office
Aberdeen Proving Ground, MD 21005-5001

Director
US Army Material Testing Directorate
Attn STEAP-MT-R, R. Harrison
Aberdeen Proving Ground, MD 21005

Commander
US Army Materiel Command
Attn AMCCN-N
5001 Eisenhower Ave
Alexandria, VA 22333-0001

US Army Night Vision & Electro-Optics
Laboratory
Attn Technical Library
Ft Belvoir, VA 22060

Commander
US Army Nuclear & Chemical Agency
Attn MONA-NU, R. Pfeffer
7500 Backlick Road
Springfield, VA 22150

Director
US Army Strategic Defense Command
Attn CSSD-H-YA, D. Stott
PO Box 1500
Huntsville, AL 35807

Commander
White Sands Missile Range
Attn STEWS-TE-AN, J. O'Kuma
Attn STEWS-TE-AN, R. Penney
White Sands Missile Range, NM 88002

Commander
Naval Electronic Systems Command
Headquarters
Attn Tech Lib
Washington, DC 20360

Commanding Officer
Naval Research Laboratory
Attn Code 6816, W. Jenkins
Washington, DC 20375-5000

Commander
Naval Surface Warfare Center
Attn Code H-23, F. Warnock
Attn Code H-23, J. E. Partak
Silver Spring, MD 20903-5000

Commanding Officer
Naval Weapons Support Center
Attn Code 6054, D. Platteter
Crane, IN 47522

Naval Weapons Evaluation Facility
Nuclear Survivability Dept
Attn R. Carroll, Jr
Kirtland Air Force Base
Albuquerque, NM 87117

Rome Air Development Center,
AF Deputy for Elec Tech
Attn W. Shedd, ESRL.G.
Hanscom Field
Bedford, MA 01731

Commander
Rome Air Development Center, AFSC
Attn RBRM, H. Dassault
Griffis AFB, NY 13441

The Phillips Laboratory, AFSC
Attn NTCTR, R. Tallon
Attn NTY, J. Ferry
Kirtland AFB, NM 87117

Sandia National Laboratories
Attn Div 2144, P. Dressendorfer
Attn Div 2147, P. Winokur
PO Box 5800
Albuquerque, NM 87185

NASA
Goddard Space Flight Center
Attn J. W. Adolphsen, Code 311
Greenbelt, MD 20771

Director
Interservice Nuclear Weapons School
Attn Tech Lib
Kirtland AFB, NM 87115

NASA
600 Independence Avenue, SW
Washington, DC 20456

Director
National Security Agency
Attn R-1, J. Hilton
FT George G. Meade, MD 20755

NASA
Sci & Tech Info Facility
Attn Tech Library
6571 Elkridge Landing Rd
Linthicum Hgts, MD 21090

BDM Corporation
Attn R. J. Antinone
1801 Randolph RD SE
Albuquerque, NM 87106

Boeing Company
Attn MS 7J-56, A. Johnston
Attn MS 2R-00, A. Measel
PO Box 3999
Seattle, WA 98124

Booz-Allen Hamilton
Attn Don Vincent
Attn Paul Deboy
4330 East-West Highway
Bethesda, MD 20814

Burruano Associates, Inc
Attn S. Burruano
PO Box 145
Harrington Park, NJ 07640

E-Systems, ECI Division
Attn C. Uber MS 19
PO Box 12248
St Petersburg, FL 33733-2248

Effects Technology, Inc
Attn Tech Lib
5383 Holister Ave
Santa Barbara, CA 93105

ESI Corp
Attn J. White
PO Box 1359
Richardson, TX 75080

General Dynamics Corporation,
Land Systems Division
Attn Nuclear Effects Group
PO Box 1800
Warren, MI 48015

General Electric Co,
Space Division
Attn J. Andrews
PO Box 8555
Philadelphia, PA 19101

Goodyear Aerospace Corp
Attn R. H. Schafer
1210 Massilltion Road
Akron, OH 44315

Goodyear Aerospace Corp,
Arizona Division
Attn Tech Lib
Litchfield Park, AZ 85340

Harris Corporation,
Government Systems Group
Attn W. A. Reed, MS 5W-446
Attn W. E. Abare, MS 20-2604
PO Box 37
Melborne, FL 32901

Honeywell Incorporated,
Aerospace Division
Attn Tech Lib
13350 US Highway 19
Clearwater, FL 33516

Honeywell SSED
Attn P. Anderson
12001 St. HWY 55
Plymouth, MN 55441

Hughes Aircraft Company,
Radar Systems Group
Attn K. Walker
PO Box 92426
Los Angeles, CA 90009

Hughes Aircraft Company, S&CG
Attn Technical Lib
PO Box 92919, Airport Station
Los Angeles, CA 90009

IBM Corporation
Attn B. Posey
9500 Goodwin Drive
Manassas, VA 22110

Interelectronics Corp
Attn M. Pintel
US Rt 303
Congers, NJ 10920

JAYCOR Corporation
Attn C. Rogers
2951 28th St., Suite 3075
Santa Monica, CA 90405-2993

Jet Propulsicn Lab
Attn Dr. C. Barnes
4800 Oak Grove Dr, M/S T-1180
Pasadena, CA 91109

Kaman Sciences
Alexandria Office
Attn DASIAC, W. Alfonte
2560 Huntington Ave
Alexandria, VA 22303

Kaman Sciences,
Center for Advanced Studies
Attn DASIAC, M. Espig
PO Drawer QQ,
816 State Street
Santa Barbara, CA 93102

Litton Systems, Inc,
Data Systems Division
Attn D. Frederick
8000 Woodley Avenue, M. S. 45-37
Van Nuys, CA 91409

Lockheed Missiles & Space Co
Attn Technical Library
PO Box 504
Sunnyvale, CA 94086

Los Alamos Tech Assoc Inc
Attn B. Johnson
PO Box 410
1650 Trinity Dr
Los Alamos, NM 87544

M.I.T. Lincoln Laboratory
Attn L. Loughlin, Librarian A-082
PO Box 73
Lexington, MA 02173

Martin Marietta Aerospace,
Orlando Division
Attn M. Griffith, Lib MP-30
Attn J. G. Simmons, MP 163
PO Box 5837
Orlando, FL 32805

Distribution (cont'd)

McDonnell Douglas Corporation
Attn Tech Lib
PO Box 516
St Louis, MO 63166

Mission Research Corporation
Attn J. Raymond
5434 Ruffin Road
San Diego, CA 92123

Mission Research Corporation
Attn R. Pease
1720 Randolph Rd, SE
Albuquerque, NM 87106-4245

Mitre Corp
Attn Library
PO Box 208
Route 62 & Middlesec Turnpike
Bedford, MA 01730

Myers & Associates
Attn D. Myers
15875 Oak Glen Ave
Morgan Hill, CA 95037

Northrop Corporation
Electronics Division
Attn S. Anderson
2301 West 120th Street
Hawthrone, CA 90250

Physitron
Attn T. G. Henderson
3325 Triana Blvd, Suite A
Huntsville, AL 35805

Physitron, Inc
Attn M. Rose
P.O. Box 27627
San Diego, CA 92128-0950

Raytheon Company
Attn H. L. Flescher
527 Boston Post Road
Sudbury, MA 01776

Raytheon Corp.
Attn G. H. Joshi, MS 12-12
Hartwell Rd
Bedford, MA 01730

Research Triangle Institute
Attn Dr. M. Simons
PO Box 12194
Research Triangle Park, NC 27709

Rockwell International Corp
Attn Technical Library
370 Miraloma Avenue
Anaheim, CA 92803

Rockwell International Corp,
Technical Information Center
Attn T. B. Yates
PO Box 92098
Los Angeles, CA90009

Rockwell/Collins
Attn A. R. Langenfeld, 137-138
855 35th Street, NE
Cedar Rapids, IA 52402

Rolm Corp
Attn H. Yue
One River Oaks Pl
San Jose, CA 95134

S-Cubed Corp
Attn Doug Willis
PO Box 85317
San Diego, CA 92138

Sylvania Electronic Systems
Attn J. A. Waldron
189 B Street
Needham Heights, MA 02194

The BDM Corporation
Attn W. Sweeney
7915 Jones Branch Drive
McLean, VA 22102

The Bendix Corporation,
Guidance Systems Division
Attn Tech Lib
Peterboro, NJ 07608

TRW
Attn A. Witteles, 134-9039
Attn Tech Info Center, S-1930
One Space Park
Redondo Beach, CA 90278

Distribution (cont'd)

US Army Laboratory Command
Attn AMSLC-DL, Dir., Corp Labs

Installation Support Activity
Attn SLCIS-CC, Legal Office

USAISC
Attn AMSLC-IM-VA, Admin Ser Br
Attn AMSLC-IM-VP, Tech Pub Br
(2 copies)

Harry Diamond Laboratories
Attn Laboratory Directors
Attn SLCHD-TL, Library (3 copies)
Attn SLCHD-TL, Library (WRF)
Attn SLCHD-NW-E, Director

Harry Diamond Laboratories (cont'd)
Attn SLCHD-NW-P, Chief
Attn SLCHD-NW-RP, Chief
Attn SLCHD-NW-TN, Chief
Attn SLCHD-NW-TS, Chief
Attn SLCHD-ST, Lab Director
Attn SLCHD-TA, Director
Attn SLCHD-TA-ES, Chief
Attn SLCHD-NW-TS, A. Abou-Auf
Attn SLCHD-NW-TS, M. Bumbaugh
(25 copies)