MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES

3 4456 0263796 6

ORNL/TM-10316

ornl

OAK RIDGE NATIONAL LABORATORY

MARTIN MARIETTA

NAVAL AVIATION INTEGRATED LOGISTICS

Technical Users Guide
Version 1.0

OAK RIDGE NATIONAL LABORATORY
CENTRAL RESEARCH LIBRARY
CIRCULATION SECTION
4500N ROOM 175
LIBRARY LOAN COPY
DO NOT TRANSFER TO ANOTHER PERSON
If you wish someone else to see this report, send in name with report and the library will arrange a loan.

R. C. DeLozier
D. A. Holder

OPERATED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road, Springfield, Virginia 22161
NTIS price codes—Printed Copy: A17 Microfiche A01

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM-10316

NAVAL AVIATION INTEGRATED LOGISTICS

Technical Users Guide
Version 1.0

R. C. DeLozier (Engineering Division)
D. A. Holder (Computing & Telecommunications)

Date Published - June 1987

Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831
operated by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
NAVAL AVIATION LOGISTICS CENTER
under Contract No. DE-AC05-840R21400

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES
3 4456 0263796 6

[BLANK PAGE]

NAIL: NAVAL AVIATION INTEGRATED LOGISTICS
Technical Users Guide
Version 1.0

Package for
Managing, Analyzing, and Forecasting
Repair and Replacement Needs for Naval Aircraft

R. C. DeLozier (Engineering Division)
D. A. Holder (Computing & Telecommunications)

CONTENTS

	Page
ACKNOWLEDGEMENTS	v
ABSTRACT	vi
1. INTRODUCTION	1
2. SYSTEM REQUIREMENTS	3
2.1 HARDWARE	3
2.2 SOFTWARE COMPATIBILITY	3
3. DESCRIPTION OF NAIL: CONTROLLER & PROGRAM STEPS	4
3.1 NAIL PROGRAM: CONTROLLER	4
3.2 FUNCTIONS	4
3.2.1 Step 1: Remove Report Headings	4
3.2.2 Step 2: Evaluate Time Between Failures	5
3.2.3 Step 3: Include R710 Data	5
3.2.4 Step 4: Regression Analysis	5
3.2.5 Step 5: Plot Data	5
3.2.6 Step 6: Distribution Analysis	5
3.2.7 Step 7: Monte Carlo Analysis	5
3.2.8 Step 8: Monthly Failure Rate	5
3.2.9 Step 9: Create Plots of Data	6
3.2.10 Step 10: Find and Count Data	6
3.2.11 Step 11: Combine Two Files	6
3.2.12 Step 12: Output Data files	6
3.2.13 Step 13: Fix R710	6
3.2.14 Step 14: Printer 132 Columns	6
3.2.15 Step 15: Printer 80 Columns	6
3.2.16 Step 16: MIR Study	6
3.2.17 Step 17: Slides for Plotter	6
3.2.18 Step 18: Setup RAM Disk	7
3.2.19 Step 19: Format Data for Regression Analysis	7
3.2.20 Step 20: Sort File	7
3.2.21 Step 25: MIR Report	7
3.2.22 Step 26: Monthly Failure Rate	7
3.2.23 Step 27: Regression Analysis	7
3.2.24 Step 31: Monte Carlo Output Example	7
3.3 FUNCTION MODIFICATIONS	8
4. DATA INPUT	9
4.1 SOURCES	9
4.2 FILE MANAGEMENT	9

	Page
5. DATA OUTPUT	11
5.1 STRUCTURE	11
5.2 FORMAT	11
6. ARTIFICIAL INTELLIGENCE APPLICATIONS	12
6.1 INTRODUCTION	12
6.2 PROLOG LANGUAGE	12
6.2.1 PROLOG & NAIL, Version 1.0	13
6.2.2 System/User Interface & Artificial Intelligence	13
6.3 NAIL: VERSION 1.0	14
6.4 NAIL: FUTURE AI ENHANCEMENTS	15
7. NAIL APPLICATIONS	16
7.1 INTRODUCTION	16
7.2 SAMPLE ANALYSIS	17
7.2.1 Selection of WUCs	17
7.2.2 WUC Analysis and Failure Projection Process	17
7.2.3 Reliability Study- Results	18
7.3 REPORT GENERATION	19
7.3.1 R710- Flight Activity and Inventory Report	19
7.3.2 R520- Individual Maintenance Action Record	19
8. PROGRAM REVISION	20
8.1 INTRODUCTION	20
8.2 ASSESSMENT	20
8.3 FIELD REVIEW	20
8.4 SCHEDULING	21
8.4.1 Introduction	21
8.4.2 Evaluation of Operational Readiness	21
8.4.3 Data Base Review	22
8.4.4 System/User Interface	22
8.4.5 Continuing Development	23
8.4.6 Field Review	23
GLOSSARY	24
REFERENCES	30

Page

APPENDIXES

Appendix A:	R710- Flight Activity and Inventory Report	A-1
Appendix B:	R520- Individual Maintenance Action Report	B-1
Appendix C:	Replacement Component Methodology	C-1
Appendix D:	Regression Analysis	D-1
Appendix E:	Master Index of Repairables (MIR)	E-1
Appendix F:	Monthly Failure Rates	F-1
Appendix G:	Monte Carlo Analysis	G-1
Appendix H:	NAIL: On-Line Tutorial	H-1
Appendix I:	Source Code Specifications: Main Programs & Modules	I-1
Appendix J:	Source Code Listings	J-1

DISTRIBUTION

ACKNOWLEDGEMENTS

The work described in this manual reached its present level of utility through the efforts of L.W. Ingalls, K. G. Kahl and V. K. Wilkinson. The authors would like to express their appreciation to Fred Fernandez, NALC, who provided valuable counsel in the course of their work.

ABSTRACT

This document summarizes the operational and analytical functions of version 1.0 of the Naval Aviation Integrated Logistic (NAIL) package. NAIL, a logistics management and analysis system, creates standardized reports and performs several categories of statistical operations in support of NAVAIR logistic analytic functions.

[BLANK PAGE]

1. INTRODUCTION

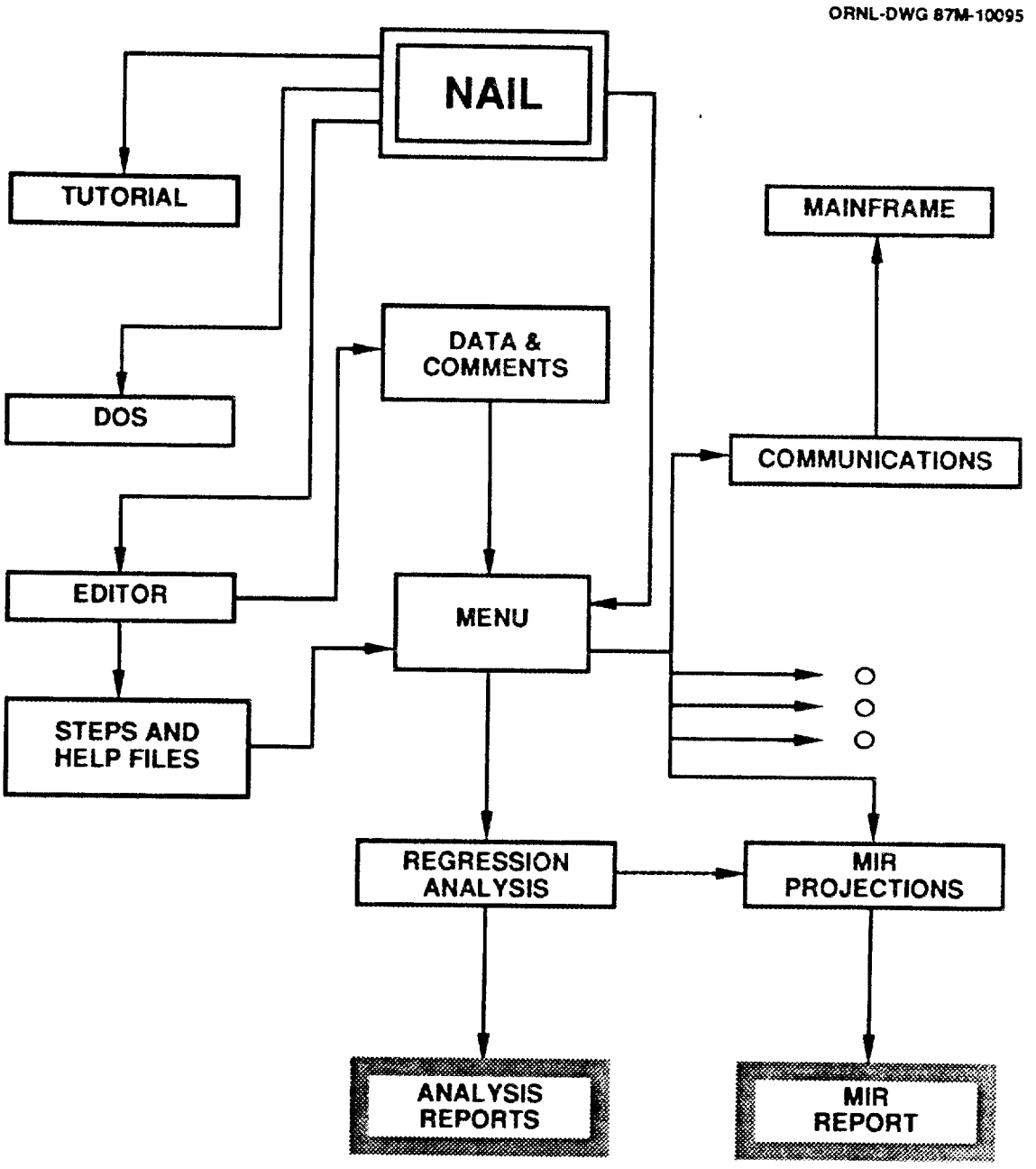
To meet the criteria for the logistics component of Reliability-Centered Maintenance (RCM) techniques, the Naval Aviation Logistics Center (NALC) requested that Oak Ridge National Laboratory (ORNL) develop an improved methodology for predicting spares provisioning. In response to the needs of NALC, ORNL personnel developed the Naval Aviation Integrated Logistics (NAIL) package.

The NAIL project may be viewed as a development of logistic support analysis (under Task 501 of MIL-STD 1388-1A):

".... to assure the achievement of specified supportability requirements, identify reasons for deviation from projects, and identify methods of correcting deficiencies and enhancing system readiness." Task 501 calls for the analysis of supportability data as it becomes available from standard supply maintenance and readiness reporting systems and from any special data collection programs implemented on systems and equipment. The results of this analysis should be used as input to Data Record B, "Item reliability (R) and maintenance (M) characteristics of 1388-2A." The standard Naval reference for reliability-centered maintenance is MIL-HDBK-266 (ref. 22).

The NAIL logistics package utilizes both historical data and current data input by the user to accomplish analyses and generate reports. By means of a simple menu program, the user may select a function from a list of options, e.g. performing statistical studies of a selected data base. A representation of the interaction among the major capabilities of the NAIL package is shown in the flow chart in Figure 1.

NAIL may be readily modified to comply with revisions in the overall Naval maintenance and reliability standards. The program architecture of NAIL may be expanded by the simple addition of codes and/or utilities to the package. The package is capable of using almost any program that will run under the DOS system. Among the modifications under development is a natural-language interface. When this interface is brought on-line, NAIL will mature into an AI system which matches the evolving logistics capabilities with the requirements of the user. The user will be able to use this future ability of NAIL without having to understand the theory of the underlying analysis.



NAIL: FLOW CHART

2. SYSTEM REQUIREMENTS

2.1 HARDWARE

The NAIL package requires a personal computer (e.g, the IBM PC or compatible) with at least 192K of memory; however, more memory is desirable. The package performs best with a hard disk and this is recommended; however, it will operate on a computer system with either a hard disk or with dual floppy disks. Any printer system should be compatible with NAIL; however, a dot matrix printer (such as an IBM or Epson) is recommended for quick draft level output. If plots are required as output, a Hewlett Packard (such as the HP 7475A) or compatible plotter is supported.

NOTE: Read/write speed is greatly enhanced by using RAM disks.

This alternate approach to disk input/output (I/O) is available if the computer system has a large amount of accessible memory. The memory can be used as a large virtual disk such as VDISK. These virtual drives can be very large and transfer data very fast. On the other hand, the necessity for using this space for some program operations, such as multi-variable regression analysis, will likely limit many large I/O data files to magnetic media on some computer configurations.

2.2 SOFTWARE COMPATIBILITY

This version of NAIL is intended for use with PC DOS or MS DOS. However, the package should install and execute correctly on almost all computer systems.

The package is compatible with most resident software; however, the ANSI device driver, if installed, may interfere with screen colors. The current version should be compatible with all RAM disks as long as there is 192K of free memory.

3. DESCRIPTION OF NAIL: CONTROLLER AND FUNCTIONS

3.1 NAIL PROGRAM: CONTROLLER

The program controller is written in the fifth generation langauge, PROLOG. This first version of the NAIL controller is constructed around a menu system, as illustrated in Figure 1. However, because it is written using the PROLOG language, it is designed to be extended into natural language applications and to interface with almost all of the PC-based analysis tools that are embedded in NAIL. The controller may also drive other tools that are developed more effectively in other languages, such as FORTRAN or PASCAL. The PROLOG source code for the controller is listed in Appendix J, as are all other source code listings.

In the present version (1.0) of NAIL, a rudimentary natural language interface is constructed to comprehend instructions or to provide HELP routines (see Appendix H, NAIL: On-Line Tutorial). Figure 1 shows how the menu-driven system uses the comments and help routines. The menu block in this figure will be modified to include more Ai capabilities. In planned future versions, this interface will be capable of translating spelling errors. Although its initial vocabulary will be small, the dictionary of the language interface will be capable of being updated. Also, if the user queries the system using language which is ambiguous to the system, the interface will question the user to resolve this ambiguity. Thus, the user may still obtain the requested information and the interface may be generated without having to include complex algorithms in the initial programming.

3.2 FUNCTIONS

The program functions, or steps, used in the NAIL package are dynamic and are very easily modified and appended. These steps are simply defined in a support file that can be modified using a NAIL utility. In figure 1, the block labeled "Steps and Help Files" contains information about each step. Currently, there is a limit of 50 steps within any one section. The following steps are those which are included in version 1.0.

NOTE: See Appendix I for a listing of source code specifications for the main programs and the modules.

3.2.1 Remove Report Headings [Step 1]

This step is normally the first step in the NAIL package. The input data file contains report headings, etc. that need to be removed before studies and analyses can be accomplished. The dates are changed to sequential numbers (1-1460 for four years) relative to the starting date of the program. This step requires input (R520 report) created outside of NAIL.

3.2.2 Evaluate Time Between Failures [Step 2]

This step simplifies the data for analysis. The aircraft tail numbers are used to find successive failures dates. When dates are identified, the difference is used to calculate the time between failures (TBF). This step requires input files created by step 1.

3.2.3 Include R710 Data [Step 3]

This step combines the step 2 file with the operational data file. The R710 data has been used to create a parameter data file (SUPERDR.SUP) that contains information on aircraft flight hours, landings, etc. This step requires input files created by steps 2, 8, and 13.

3.2.4 Regression Analysis [Step 4]

This step performs linear regression analysis on the file generated by step 3. The correlation equation is assumed to be linear and each parameter can be omitted to determine its effect. This program also can generate plots of data. This step requires input files created by step 19. (See Appendix D for sample Regression Analysis.)

3.2.5 Plot Data [Step 5]

This step generates plots of data created by other steps. The plot can display data in the form of bars, symbols, or lines using an HP plotter, or compatible.

3.2.6 Distribution Analysis [Step 6]

This step evaluates and fits the failure data distribution function. The program also creates a temporary data file for use in Monte Carlo simulation of part inventory requirements. This step requires input files created by step 4.

3.2.7 Monte Carlo Analysis [Step 7]

This step does a simulation of the part inventory using the observed distribution function. This simulation indicates the stock required to reduce the probability of zero inventory to an acceptable confidence level. This step requires input files created by steps 6 and 8.

3.2.8 Monthly Failure Rate [Step 8]

This step does an initial evaluation of the number of failures per month for a given time interval. The mean time between failures (MTBF) and other values are estimated. The monthly failure rate is fit to a Weibull projection equation. This step requires input files created by step 2.

3.2.9 Create Plots of Data [Step 9]

This step is used to create plot files using data input from the keyboard and/or files specified by the user. Several equations can be fitted to the data and plotted as a line along with the input data.

3.2.10 Find & Count Data [Step 10]

This program evaluates the number of occurrences of data in given columns of a data set. For example, this program is used to count the number of repairs at O, I, or D level.

3.2.11 Combine Two Files [Step 11]

This program is used for combining two data sets. For example, the maximum time interval for the R520 report is two years; this program allows two such data sets to be combined into a four year set.

3.2.12 Output Data Files [Step 12]

This program outputs data files to the printer along with column numbers. The output is used to find the column numbers of data fields for input to data management programs.

3.2.13 Fix R710 [Step 13]

This program is used to reduce the R710 report to a simplified data set. This step requires input files created outside of NAIL.

3.2.14 Printer 132 Columns [Step 14]

This step will change the output of Epson-type printers to 132 columns per line. This is required for most output in the NAIL system.

3.2.15 Printer 80 Columns [Step 15]

This step will change the printer output back to 80 column if needed.

3.2.16 MIR Study [Step 16]

This analysis is used to project the number of removals and repairs at the O, I and D levels using a data set created by the step 2 activity.

3.2.17 Slides for Plotter [Step 17]

This step generates data files that can be plotted as text files. The HP plotter can be used to output slides with this utility.

3.2.18 Setup RAM Disk [Step 18]

This step assumes the use of SUPERDRV (AST) but the user can edit the QDELOGIC file and change SUPERDRV to any other procedure.

3.2.19 Format Data for Regression Analysis [Step 19]

This step changes the output file created by Step 2 to the compressed format required for input to step 4. This step can be used to remove time between failures having zero values.

3.2.20 Sort File [Step 20]

This step is used to sort data files based on specified columns. If the R520 report is not sorted, this procedure can be used after step 1.

3.2.21 MIR Report [Step 25]

The Master Index of Repairables (MIR) Report is produced from the R710 and R520 data base during this step. (See Appendix E for a sample MIR Report.)

3.2.22 Monthly Failure Rate [Step 26]

During this step, a report on the Monthly Failure Rate of a given set of parts is created from the R710 and R520. (See Appendix F for a sample Monthly Failure Rate report.)

3.2.23 Regression Analysis [Step 27]

A multivariable regression analysis is run utilizing the R710 and R520 reports at this program step.

3.2.24 Monte Carlo Output Example [Step 31]

The Monte Carlo analysis is an repetitive and lengthy analysis that consists of many steps. This analysis projects the number of repairs at the O. I, and D level. The example shows the results of two such analyses. (See Appendix G for a discussion and an example of Monte Carlo analysis.)

3.3 FUNCTION MODIFICATIONS

The previous function steps are defined in a file named DELOGIC. This file contains all information required to accomplish a function plus identification and help messages. In Figure 1 this file is represented by the block labeled "Steps and Help Files." The DELOGIC file can be modified using the controller utility provided or any word processor. The format for an entry is as follows:

AAAAA^BBBBB | C1 | C2...C_n |

where A's represent a text identifier for a given operation, B's represent a help message or elaboration on the function, and C's denote distinct DOS operations, utilities, or program calls. The identifier and help text are separated by the "^" character. The help text is separated from the distinct operation with the "|" character. The main limitation is that the total set of characters for one operation is no more than 480 characters long, total. There is no limit on the number of operations (C1, C2,etc.) beyond the 480 character limit. This restriction may be overcome by simply defining a BAT file with any number of operations and calling this BAT file as one operation.

4. DATA INPUT

4.1 SOURCES

The two major sources of data input to the NAIL System are the Individual Maintenance Action Record (R520 Report) and the Flight Activity and Inventory Report (R710 Report). The Individual Maintenance Action Record describes the result of particular maintenance actions. The report indicates when a problem was discovered and the action taken. The Flight Activity and Inventory Report provides monthly totals of aircraft inventory and utilization activity levels. Both reports are generated using the Analytical Maintenance Program Analysis Support System (AMPAS).

AMPAS is a computer system designed to provide analysis procedures and techniques on maintenance requirements of each aircraft. The AMPAS data base contains information on inventory data and support equipment for aircraft, aircraft utilization, capability data for aircraft, and maintenance data for all equipment. The primary input data file from which the majority of the AMPAS application programs are executed is a Naval maintenance data file which is maintained by the Naval Aviation Logistics Center. This file contains maintenance and selected readiness data on Naval aircraft, engines, and Ground Support Equipment. Information concerning aircraft readiness and utilization is obtained from the Flight file. This file is also maintained by the Naval Aviation Logistics Center. The Flight file data is arranged chronologically in individual aircraft records and is updated monthly.

Before using the NAIL System the user must generate the R520 report and the R710 report for a selected WUC and date range. In order for the NAIL system to recognize these reports, they must be downloaded to the PC using a communication program (e.g., PC-Talk) and stored on the PC using the file extension .DAT.

4.2 FILE MANAGEMENT

When the user enters the NAIL System, the system will request the names of the default disk drive and directory on which the input and output data reside. The assumed default drive and directory are displayed on the screen and may be overwritten or accepted. Default file extensions are also used by the NAIL System to aid in the management of files.

The NAIL System associates files with a file extension of .DAT as an R520 Report or a R710 Report. The reports must be downloaded to the PC with this file extension in order for NAIL to recognize them. The NAIL system will append the appropriate standard file extension for any files created by the NAIL system. The table below indicates the file extensions used by NAIL and the NAIL Step which creates the output file. Both input file and output file extension are shown.

STEP	INPUT FILE EXTENSION	OUTPUT FILE EXTENSION
1	.DAT	.DTA
2	.DTA	.DTB
3	.DTB	.DTC
19	.DTC	.DTD
4	.DTD	.R04
8	.DTB	.R08
16	.DTB	.R16

Using standard file extensions allows the NAIL System to produce a directory listing of files that can be selected for input. When NAIL prompts for a file name, the directory listing can be viewed by pressing the RETURN key. A file can also be selected from the directory listing by positioning the cursor on the appropriate file name and pressing the Esc key.

When the user executes a function, or step, of NAIL a comment screen will appear. The NAIL System will display comments about the step that was executed and the output file created. The user may edit or add to the comments created by the NAIL System.

The comments entered may be helpful later when selecting a file from the directory listing. The user may view the comments entered for a particular file from the directory listing by positioning the cursor on the file name and pressing the F1 key. The comments are also used to reference the input file used in generating a report. The input file comments are displayed on the first page of the report as a reference guide.

5. DATA OUTPUT

5.1 STRUCTURE

When the R520 report and the R710 report are processed by the NAIL System, various data sets are created. The following figures from Appendix H: NAIL On-Line Tutorial indicate the proper sequence of steps necessary to generate the MIR Report, the Monthly Failure Report, the Regression Analysis, and the various data sets that are created as each step is executed.

Figure 4 shows the steps necessary to produce the MIR Report and the sequence to execute the steps.

Figure 5 indicates the steps necessary to produce the Monthly Failure Report.

Figure 6 shows the steps used to generate the Regression Analysis.

To produce each of the reports, the steps must be executed in the sequence illustrated in the above flow diagrams. As the data sets are created from generating one report, the steps used to generate these data sets may be omitted when generating a new report, unless a new R520 or R710 report is being used. (See Appendix A: R710-Flight Activity and Inventory Report and Appendix B: R520-Individual Maintenance Action Report to view the structure of each data set created.)

5.2 FORMAT

The NAIL System utilizes data from the R520 report, the R710 report, and user input to generate an MIR report, a Monthly Failure report, and the Regression Analysis. Refer to the following Appendixes for the format of each report:

Appendix D: Regression Analysis
Appendix E: Master Index of Repairables (MIR)
Appendix F: Monthly Failure Rates.

6. ARTIFICIAL INTELLIGENCE APPLICATIONS

6.1 INTRODUCTION

The future use of Artificial Intelligence (AI) programming in the NAIL package is intended to provide the user with a natural language interface capability. Through this interface, users can communicate their logistics needs in a direct manner. Instead of entering a complex sequence of specifications into the system, the user simply types in a natural language question or a command. With these enhancements, NAIL would comprehend the request and either perform actions in support of the request, or ask the reader to enter other questions or to respond to a question.

One of the goals of AI programming is the creation of a computer system which understands natural language patterns (albeit restricted to limited subject matter) and fashions responses based upon human reasoning and learning processes. This "thinking machine" has been actively imagined for generations. A machine which will understand human language and replicate human thought has often been depicted in literary and speculative writing. However, since the electronic revolution the creation of machine intelligence appears to be more of an eventuality than a fantasy.

Although the enigmatic HAL in the film of Arthur C. Clarke's *2001*, and the fussy, gold-plated C3PO in the *Star Wars* trilogy, remain special effects, electronic replication of human thought processes may soon be a reality. Some AI researchers predict that, within the next few decades, man and machine may co-exist as thinking beings.

6.2 PROLOG LANGUAGE

The NAIL development plan calls for the phased production of a system that will serve as a major logistic planning tool for NAVAIR while having a very user friendly interface. The decision to develop the package using the PROLOG language is concurrent with this plan. This langauge is one which is capable of supporting a portable natural language interface. Additionally, it has many strong AI capabilities.

Many of the programming languages which have been in use since the early days of computers are *procedural*. FORTRAN, COBOL, and Pascal are three of the better known languages of this type. In these languages the programmer must determine and include every step needed to solve a problem or perform a given operation. The programming can become very complex and resource-consuming when each step needs to be described. This also means that the range of responses for a given situation is limited and that the program is unable to learn any new information without reprogramming.

The advent of a new type of programming language--the *declarative* language--has created new ways for computers to solve problems, to learn, and to communicate with the user. A declarative language is based upon a set of rules and facts which are used to solve problems and to comprehend user requests. Thus, programs in

these types of languages do not have to contain each possible step in the procedure or each probable response. Like Sherlock Holmes, Miss Marple, or Lew Archer, the program uses deductive reasoning to solve problems. Responses are entered into the language interface of the program, and the program applies a set of facts and rules to solve the problem. (Turbo Owner's Manual, 1)

A declarative programming language may learn new sets of rules and new facts. These rules and facts may be entered into the system at the level of the user interface. Declarative languages may thus be used to create programs capable of learning and of responding to "first-time" situations and problems.

Because they are "rule-driven," declarative languages may interface with a user or another system through any form of communication which is described by rules. Natural language is formulated upon sets of rules for constructing and comprehending sentences, and for deducing actions and implications from linked groupings of sentences. Thus, declarative language programs have the capability to provide natural language interface. The user enters a request in natural language instead of entering a series of machine commands.

6.2.1 PROLOG & NAIL, Version 1.0

A declarative language used for NAIL applications is Turbo PROLOG. This language package is utilized as a compiler for the NAIL controller. The compiler is a program which will change a higher level language into machine instructions. In the NAIL package, the controller was compiled using Turbo PROLOG. This programming language creates executable files with the suffix EXE. This suffix designates a file that can be rapidly executed underneath the operating system (i.e., DOS) without the use of support software.

Within this overall logistics management program, PROLOG has been used to create a controller and editor for specified analytical definitions. The NAIL controller calls, combines, and runs individual or sequential operations. For example, when the user initiates step 3 of NAIL, the controller combines the file created during step 2 with the operational data file. This sequential operation is performed without additional input from the user.

PROLOG was also used to create a program which enables the user to access the tutorial, use the HELP system, and execute DOS commands while in the NAIL routine. Through the use of the language, the user accessibility of many existing PC capabilities has been enhanced. Future planned developments using PROLOG-based programming allow the user to communicate in natural language and simplify the commands needed to identify and execute required analytical routines.

6.2.2 System/User Interface & Artificial Intelligence

Clearly, the interface between the user and the NAIL system is important to the overall effectiveness of this logistics management package. This interface must be user-friendly and meet the needs of all users, whether technical or non-technical. Presently, prompts, explanations, "help" screens, and a tutorial are placed within the program package to facilitate its application by the user. These guideposts help the user to understand the total systems operation, direct the user through a given step in the NAIL package, and provide guidelines for making operational inferences from the analytical data generated by NAIL.

Present efforts to establish the optimum user-friendly interface are concentrating upon using PROLOG to create a natural language interface founded upon Artificial Intelligence (AI) concepts. An AI-based system would enable the user to direct sequences of operations using natural language formats rather than having to enter sequences of specifications into the system. Instead of entering a chain of machine commands to produce information, the user could use natural language to make requests of the system. For example, in place of a sequence of command codes, the user could type in : " Print-out the latest R520 on WUC 42111 for [aircraft identification number]."

This natural language interface would also recognize and correct errors (e.g., spelling) made during query and data entry. The vocabulary in the initial versions of NAIL will be deliberately simple, though adequate for the assigned tasks. In future versions, additions to the vocabulary may be made by the users, or inferred by the system. In this manner, changes in terminology and site-specific usages may be included in the NAIL vocabulary . The dictionary available to the NAIL package would be continually upgraded by the AI system as part of the user/NAIL verbal interaction.

A user interface made possible by the PROLOG language could eventually operate at a very high level of understanding. At this level of AI, the user could request the system to both perform a series of intricate analyses and to inform the user of the nature of the requested logistics operations (e.g. "I want to know if I should order replacements for the P-3 IFF transmitter. Tell me what I need to know to make and to implement this decision."; or, "What needs to be done on the P-3 aircraft?").

The planned inclusion of AI precepts in the NAIL interface should produce a powerful tool for logistics operations and management, and provide access to a large number of other capabilities. An AI-driven interface system could also enable NAIL system users to more easily access the extensive Naval data bases using the relational data base capabilities used in the language. AI programming could also support voice-synthesizer technology to facilitate the use of NAIL by the visually-impaired. Therefore, research and programming relative to developing PROLOG-based user interfaces is continuing. The primary goal of the project is to ensure that future versions of NAIL will be both simpler to use and able to make complex logistical management decisions.

6.3 NAIL: VERSION 1.0

At present, technology and programming are not at the level of parallel processing and integration required for full-scale modeling of human thought, programming tools (e.g. PROLOG) have been implemented which allow natural language to be used for man-machine communications. This type of interface is available to the user of NAIL.

Users of the current version (1.0) of NAIL may interact with the program package by entering simple sentences at the designated level or step. For example, in place of a sequence of command codes, the user asks the program to execute a specified step and to print this information.

6.4 NAIL: FUTURE AI ENHANCEMENTS

The AI interface presently installed in NAIL will be augmented in future versions. Work will concentrate on the addition of modifications that will support the execution of increasingly complex operations. The goal is to reach a level of AI in which the user could request the system to both perform a series of intricate analyses and to provide information on the nature of the requested logistics operation.

Enhanced versions of NAIL will be able to discern the user's requirements to a more detailed level and to directly prompt the user concerning site-specific logistics specifications. The inclusion of current and advanced AI concepts into the NAIL user interface will produce a powerful tool for logistics management.

7. NAIL APPLICATIONS

7.1 INTRODUCTION

NAIL provides a methodology for predicting and managing spare parts. This logistics package, which is capable of modeling multiple variables, provides the user with an analytical and verifiable basis for supply support allocations. By this means, the confidence level between predicted and observed logistics values may be projected, understood, and possibly improved. Through the effective use of NAIL, the readiness of fleet systems could be enhanced, and more efficient planning and budgeting measures within the Naval Air Systems Command (NAVAIR) could be established and promoted.

At the depot level, NAIL provides the user with an historical and analytical view of the spares provisioning requirements subject to user-designated constraints. The package will create specified reports, perform logistics analyses, and access given Naval data bases. Using these reports, logistics personnel may project spares provisioning requirements and obtain estimates of probable failures for selected parts or systems. Individual component or entire weapon system reliability factors may also be studied through the application of NAIL.

An attempt has been made to ensure that flexible output formats are readily available. Selection and sort requirements are determined by user application. The ability to select exactly the problem equipment of interest and to sort at equipment levels of interest enables the analyst to focus on the individual maintenance records which underlie aggregate equipment problems. The capability to vary selections and sorts enables the user to look at maintenance repair and replacement history from any perspective and equipment level.

7.2 SAMPLE ANALYSIS

The following narrative is an example of an analysis performed on P-3C type/model aircraft data in support of NALC logistics management objectives. Data from this study was used to define and develop some of the basic logistics management parameters for the NAIL package.

Statistical analysis of data from the NALC P-3C data bases was performed using standard techniques, such as those presented in *Quality Control and Industrial Statistics* (Duncan, 1974) and *Probability and Statistics in Engineering and Management Science* (Hines & Montgomery, 1972).

7.2.1 Selection of WUCs

For this study, two WUCs were selected: 65341- IFF Receiver Transmitter; 73668-319 Tape Transport. The base criteria for parts selection were parts which had an adverse effect upon the level of fleet readiness primarily due to unscheduled maintenance.

The primary P-3C data sets for the WUCs received from NALC were generated using the Analytical Maintenance Program Analysis Support System (AMPAS). Failure data was transmitted using the form 520 reporting format. These data were sorted by organization code, bureau number, and job control number date. Records involving the WUCs specified for analysis were selected and tabulated. A corresponding operational data set was assembled and transmitted based on the form 710 reporting format. This data set contained information on system level activities such as flight hours, and numbers of takeoffs and landings.

7.2.2 WUC Analysis and Failure Projection Process

After the maintenance and operational data for the two WUCs, and a corresponding set of operational data, had been obtained from the NALC data base, a procedure for analyses and failure projections (see Appendix C: Replacement Component Methodology) was used to estimate component failure and replacement requirements. The first step in this process is to analyze, sort, and correct obvious errors in the data. The process of analyzing, sorting, and correcting is an iterative process whereby the Time-Between-Failures (TBF) is estimated and then analyzed to verify that the results are reasonable. This analysis frequently required that the input data be screened more carefully and errors corrected where possible. After the iterative process was completed, TBF was estimated by regression analysis.

The regression analysis determined the time between failures as a function of several independent variables, such as time, flight hours,landings, takeoffs, etc. The regression analysis identified possible correctable independent variables for the entire P-3 fleet. An analysis of failures at each P-3 organization was made to identify those squadrons which showed either very good or very bad maintenance histories. Expert opinion was also included in this analysis. The primary source of expert opinion for the selected P-3 components was maintenance personnel from the Patuxent River P-3 maintenance squadron.

After the regression analysis was complete, the type of failure distribution was estimated by examining the frequency distributions of the individual parameters (i.e., time-between-failures, flight hours, calendar time). This frequency distribution analysis determines the type of distribution, such as exponential, normal, Weibull, or other standard reliability distributions. Given the type of failure distribution, the probability density function parameters such as sigma, mu, beta, or lambda were estimated. Once the type of failure distribution was established, unreliability was estimated. From this unreliability estimate, the appropriate failure rate for the component was established. With an estimate of failure rate, the expected number of failures over a given time period or number of flight hours was projected.

The last step in this analysis was to estimate the replacement components required at the operational, intermediate, and depot level using the numbers of failures projected. It was not possible to fully accomplish this last step due to lack of depot-level data. These data were unavailable from the NALC data base for the two P-3 components used in the study. The depot level is where the component is ultimately repaired or replaced. Thus, only estimates of replacement or repair activity at the lower levels could be performed based on actual data. The programs will attempt to estimate these numbers. In this latter case the numbers will be "0" due to lack of sufficient data.

7.2.3 Reliability Study- Results

The results of studies on P-3 component reliabilities using standard reliability engineering techniques resulted in specific R&M recommendations in eight areas: technology-based reliability and maintainability programs; reliability and maintainability demonstration programs; full-scale engineering development (FSED) planning and analysis; reliability and maintainability standards; FSED management and awareness of reliability and maintainability; new systems maturity; collection and use of reliability and maintainability data; and reliability and maintainability training.

A conventional measure of the useful lifetime of an aircraft is generally recognized to be flight-hours. The linear relationship that is presumed to exist between flight-hours and failures is also widely used, although the analysis indicated that there is little basis for the acceptance of this correlation other than the assumption that many parts of the aircraft are characterized by constant failure rates (failures/flight-hour). For some components, this assumption is valid; however, for many "problem" components this correlation is small. One technique in NAIL is to look for time-dependent correlation of failures. When failures are increasing with time without a corresponding increase in parameters such as flight hours, then further investigation is required.

During the periods of early life and later life, some parts exhibit rapidly changing failure rates. Thus, for this class of parts, this assumption of direct correlation between failures and flight hours does not hold. Additionally, there may be incremental or alternate factors which might modify this view of failure as being time-dependent. During the preliminary phase of the development of the NAIL package, these types of contributing factors were investigated and additional analytical strategies for NALC were proposed for incorporation into the package.

The current assumption used in NAIL is that additional investigation on a part is necessary if time-dependent correlation is observed. Spares Requirements for a part that has increasing or decreasing failure rates cannot be projected easily without understanding the time-dependent phenomena.

7.3 REPORT GENERATION

The NAIL package utilizes information from user input and from selected data bases to create and update reports. The two major reports which NAIL both uses as sources of information and generates are the R710 and the R520.

7.3.1 R710- Flight Activity and Inventory Report

This operational report provides squadron-level monthly totals of aircraft inventory and utilization activity levels in terms of flight hours, number of sorties, average number of aircraft in the reporting inventory, and hours in readiness reporting status. The maximum time span covered by this report is 24 months. (See Appendix A for sample report.)

7.3.2 R520- Individual Maintenance Action Report

This record provides a detailed description of the results of particular maintenance actions. This description includes qualitative information: e.g. when discovered, action taken, malfunction description codes, detailed equipment identification (Type/Model/Series), aircraft bureau number, Work Unit Code (WUC), Part Number, and Serial Number. The report also includes the quantitative impact of individual maintenance actions, such as man-hours, aborts, degraded time, and mishaps. These individual maintenance occurrences are the raw data which are consolidated, monitored, and used as source data in other programs. This data is used for detailed analyses and as Standard Requirements Code (SRC) unscreened input data, and for research into interrelation of problem categories. (See Appendix B for sample record.)

8. PROGRAM REVISION

8.1 INTRODUCTION

The NAIL package has been designed to undergo modifications corollary to changes in the logistic needs of NAVAIR and to respond to information from data reassessments and field. These modifications to the NAIL package will include all aspects, from reliability theory and AI-based programming, to equipment configuration and enhancements. The following sections discuss theses aspects of the continuing evolution and improvement of the NAIL package.

8.2 ASSESSMENT

The procedures used in the NAIL system will be assessed through entering historical data which is applied to selected aircraft WUCs. Projections will be made and compared to data which is not used to support the projections. From these simulations and discussions with knowledgeable individuals, the project staff will develop and modify analytical techniques that will extend and improve the capabilities of NAIL. A continuing effort will be directed toward identifying and including NAVAIR reports that support the logistic effort. These reports will be based on existing analyses. New capabilities will be developed and included to support logistic needs.

8.3 FIELD REVIEW

A final component of the NAIL development methodology is the on-going process of gathering and incorporating comments and suggestions from field testing of the NAIL package. During and after each field test, users are queried extensively. They are asked to comment on such program topics as set-up of the package, comprehension of instructions, user input of information, and applicability of the present version of NAIL to Naval logistics requirements. Follow-up contacts (questionnaires and interviews) with field personnel are also made in order to gather additional information and to collect user queries. A user comment sheet is also incorporated into the NAIL package. This comment sheet will be sent directly to the development group at ORNL. These field and user data are compiled and collated at designated intervals.

During field testing, the hardware configurations utilized by the Navy will also be reviewed for compatibility by ORNL project personnel. This compatibility review will ascertain whether the as-designed system will run on all potential Naval system configurations. Modifications to the NAIL package for conformity to Naval hardware will be made, if required.

Information from the user review process is utilized as the basis for test reports and recommendations. This field source data is then used by ORNL to direct modifications to the NAIL program package. Based upon this input, we will establish the new report needs and priorities. These reports will be included as an orderly part of the NAIL development process; however, if required, they can be incorporated into the package on an accelerated schedule.

8.4 REVISION SCHEDULE

8.4.1 Introduction

The following discussion concerns the future direction of five components of the NAIL package development program. These five areas are considered to be central to the formulation and installation of a field-use logistics system incorporating NAVAIR specifications for usefulness and adaptability. The five components are: Evaluation of Operational Readiness; Data Base Review; System/User Interface; Continuing Development; and Field Review.

8.4.2 Evaluation of Operational Readiness

The NAIL package can currently be used to calculate the operational readiness of aircraft and/or support systems as a function of performance parameters based on varying scenarios. This capability could permit the evaluation of the effectiveness of systems during wartime conditions given a table of minimum requirements and maximum capabilities for a particular aircraft (such as the F-14). These min/max tables can be included in the NAIL structure very easily and would result in a new computational option: the ability to evaluate operational readiness for a system, a squadron, or a fleet under wartime or conflict conditions.

The NAIL package also permits the statistical confidence associated with the calculations of systems readiness to be evaluated as part of the analysis. The min\max tables can be constructed based on projected capabilities, historical data, or information obtained from flight line personnel. For example, the proposed package could simultaneously report both the number of aircraft that are operational under wartime conditions and statistical information, such as the attendant standard deviation, associated with that projection. Information of this type would permit strategic decisions to be made based upon the probability of success.

This procedure is very flexible and can be extended to many areas of performance measurement. For example, the operational and material costs can be evaluated using a similar procedure. A table or matrix of component costs plus labor-related costs could be added to give cost comparisons of various options. This facility, coupled with the operational readiness option, could be used to maximize readiness while minimizing costs.

By including the capability to evaluate operational readiness for components, systems, squadrons, and other areas of performance measurement the NAIL system would assist decision-making during states of both full and reduced operational capabilities. Thus, efforts to perfect this feature of the NAIL package will be a major aspect of the continuing program.

8.4.3 Data Base Review

The projection capabilities of NAIL are founded upon the historical data base. This data is used to test the logistics capabilities of NAIL and, therefore, is a crucial component of the development process. This historical information sets the baseline for the statistical analyses and inferential studies NAIL applies to the information which is input by the user to produce all required reports. The NAVAIR data base will continue to be studied and screened so that information relevant to logistic operations may be utilized to update the NAIL package.

8.4.4 System/User Interface

The interface between the user and the NAIL system is important to the overall effectiveness of this logistic management package. This interface must be user-friendly and meet the needs of all users, whether technical or non-technical. Prompts, explanations, "help" screens, and a tutorial are placed within the NAIL program package to facilitate its application by the user. These guideposts help the user to understand the total systems operation, direct the user through a given step in the NAIL package, and provide guidelines for making operational inferences from the analytical data generated by NAIL.

The optimum user-friendly interface incorporates a natural language system founded upon Artificial Intelligence (AI) concepts. An AI-based system enables the user to direct sequences of operations using natural language formats rather than having to enter sequences of specifications into the system. Instead of entering a chain of machine commands to produce information, natural language is used to make requests of the system. For example, in place of a sequence of command codes, the user could type in : " Print-out the latest R520 on WUC 42111 for [aircraft identification number]."

This natural language interface will recognize and correct errors (e.g. misspellings, incorrect WUC) made during query and data entry. Errors in command entry and other user errors will also be flagged. The vocabulary in NAIL will be deliberately simple, though adequate for the assigned tasks. Additions to this vocabulary may be made by the users. In this manner, changes in terminology and site-specific usages may be included in the NAIL vocabulary . The dictionary available to the NAIL package will be amended by the AI system as part of the user/NAIL verbal interaction.

This interface will eventually operate at a very high level of understanding. At this level of AI, the user could request the system to both perform a series of intricate analyses and to inform the user of the nature of the requested logistics operations (e.g. "I want to know if I should order replacements for the P-3 IFF transmitter. Tell me what I need to know to make and to implement this decision.")

The inclusion of AI precepts in the NAIL interface will produce a powerful tool for logistics operations and management. An AI-driven interface system would enable NAIL system users to more easily access the extensive Naval data bases. AI programming will also support voice-synthesizer technology to facilitate the use of NAIL by the visually-impaired.

8.4.5 Continuing Development

Task activities in support of the development of the NAIL package are proceeding. On-going project operations include: development of technical documentation; programming numerous user and installation procedures; and experimental running of the NAIL system on selected aircraft Work Unit Codes (WUCs) on typical examples of depot and field computer equipment configurations.

Programming activities in support of analysis of logistics parameters within NAIL will continue. The conversion of plot routines, improving the on-line "Help" system, installation procedures, communications, and input/output routines are among the tools under periodic review. This programming architecture will be both supportive of all as-planned NAIL functions and modifiable to sustain amendments to the logistics package.

The procedures used in the NAIL system will continue to be assessed through inputting historical data which is applied to selected aircraft WUCs. Projections are made and compared to data which is not used to support the projections. From these simulations, analytical techniques may be modified and developed that will improve the capabilities of NAIL.

8.4.6 Field Review

The gathering and incorporation of comments and suggestions from field testing of the NAIL package will continue to be an important aspect of NAIL. During and after each field test, users are queried extensively. They are asked to comment on such program topics as set-up of the package, comprehension of instructions, user input of information, and applicability of the present version of NAIL to Naval logistics requirements. Follow-up contacts (questionnaires and interviews) with field personnel will also be made in order to gather additional information and to collect user queries. A user comment sheet is incorporated into the NAIL package. This comment sheet will be sent directly to the development group at ORNL. These field and user data are compiled and collated at designated intervals.

Information from the user review process will be utilized as the basis for test reports and recommendations. This field source data will be used to direct modifications to the NAIL program package. Based upon this input, new report needs and priorities will be established. These reports will be included as an orderly part of the NAIL development process.

GLOSSARY

ACRONYMS

AI (Artificial Intelligence)

AMPAS (Analytical Maintenance Program Analysis Support)

AMPL/LM (Program Manager for Logistics/Logistics Manager)

COC (Coefficient of Correlation)

DOS (Disk Operating System)

FMC (Full Mission Capable)

FRA (Fleet Repairable Assets)

FSED (Full-scale Engineering Development)

HDL (Harry Diamond Laboratory)

I/O (Input-Output)

LSA (Logistics Support Analysis)

MIR (Master Index of Repairables)

MS (Microsoft^{R})

NALC (Naval Aviation Logistics Center)

NALCM (Naval Air Logistics Center Managers)

NALDA (Naval Aviation Logistic Data Analysis)

NARF (Naval Air Rework Facility)

NAVAIR (Naval Air Systems Command)

NMC (Not Mission Capable)

O,I,D (Organization, Intermediate and Depot levels)

ORNL (Oak Ridge National Laboratory)

PC (Personal Computer)

PM (Preventive Maintenance)

PMC (Partial Mission Capable)

RCM (Reliability-Centered Maintenance)

RIP (Readiness Improvement Program)

SAS (Statistical Analysis System)

SRC (Standard Requirements Code)

TBF (Time-Between-Failures)

TBMA (Time-Between-Maintenance-Actions)

VIDS/MAF (a maintenance action form)

WUC (Work Unit Code)

DEFINITIONS

ARTIFICIAL INTELLIGENCE- a term which may be applied to the type of intelligent program which controls NAIL functions; an intelligent program exhibits behavior similar to that of a human when confronted with a similar problem. It is not necessary that the program actually solve, or attempt to solve, the problem in the same way as a human. (Schildt, 1987)

AVAILABILITY- a measure of the degree to which an item is in an operable and committable state at the start of a mission when the mission is called, or at an unknown (random) time. (MIL-STD-721-C)

BIVARIATE REGRESSION ANALYSIS- determination of the mean expected value of one dependent random variable relative to another. (Hines & Montgomery, 1972)

CONFIDENCE LEVEL- a relative measure related to the confidence interval. The probability of the predicted outcome varies directly with the value of this level. (Hines & Montgomery, 1972)

CONSTANT FAILURE RATE- the probability of failure does not vary with time. (Hines & Montgomery, 1972)

CRITICALITY- a relative measure of the consequence of a failure mode and its frequency of occurrence. (MIL-STD-721-C)

DEPENDABILITY- a measure of the degree to which an item is operable and capable of performing its required function at any (random) time during a specified mission profile, given item availability at the start of the mission. (Item state during a mission includes the combined effects of the mission-related system R&M parameters, but excludes non-mission time.) (MIL-STD-721-C)

ENVIRONMENT- the aggregate of all external and internal conditions (such as temperature, humidity, radiation, magnetic and electric fields, shock vibration, etc.) either natural or manmade, or self-induced that influence the form, performance, reliability, or survival of an item. (MIL-STD-721-C)

FAILURE- the event, or inoperable state, in which any item or part of an item does not, or would not, perform as previously specified. (MIL-STD-721-C)

FAILURE MECHANISM- the physical, chemical, electrical, thermal, or other process which results in failure. (MIL-STD-721-C)

FAILURE MODE- the consequence of the mechanism through which the failure occurs, i.e. short, open, fracture, excessive wear. (MIL-STD-721-C)

FAILURE MODE AND EFFECTS ANALYSIS (FMEA)- a procedure by which each potential failure mode in a system is analyzed to determine the results or effects thereof on the system and to classify each potential failure mode according to its severity. (MIL-STD-721-C)

FAILURE, RANDOM- failure whose occurrence is predictable only in a probabilistic or statistical sense. This applies to all distributions. (MIL-STD-721-C)

FAILURE RATE- the total number of failures within an item population, divided by the total number of life units expended by that population, during a particular measurement interval under stated conditions. (MIL-STD-721-C)

HISTOGRAM- a representation of a frequency distribution by means of rectangles whose widths represent class intervals and whose heights represent corresponding frequencies. (Hines & Montgomery, 1972)

INHERENT R & M VALUE- a measure of reliability or maintainability that includes only the effects of an item design and its application, and assumes an ideal operation and support environment. (MIL-STD-721-C)

KURTOSIS- a measure of the occurrences at the mean relative to those along the tails of the distribution. [A normal curve has zero kurtosis.] It is the fourth moment about the mean. (Hines & Montgomery, 1972)

MAINTAINABILITY- the measure of the ability of an item to be retained in or restored to a specified condition when maintenance is performed by personnel having specified skill levels, using prescribed procedures and resources, at each prescribed level of maintenance and repair. (MIL-STD-721-C)

MAINTENANCE- all actions necessary for retaining an item in or restoring it to a specified condition. (MIL-STD-721-C)

MAINTENANCE, CORRECTIVE- all actions performed as a result of failure, to restore an item to a specified condition. Corrective maintenance can include any and all of the following steps: Localization, Isolation, Disassembly, Interchange, Reassembly, Alignment, and Checkout. (MIL-STD-721-C)

MAINTENANCE, PREVENTIVE- all actions performed in an attempt to retain an item in specified condition by providing systematic inspection, detection, and prevention of incipient failures. (MIL-STD-721-C)

MEAN-TIME-BETWEEN-DEMANDS (MTDB)- a measure of the system reliability parameter related to demands for logistic support: The total number of system life units divided by the total number of item demands on the supply system during a stated period of time, e.g. Shop Replaceable Unit (SRU), Weapons Replaceable Unit (WRU), Line Replacement Unit (LRU), and Shop Replaceable Assembly (SRA). (MIL-STD-721-C)

MEAN-TIME-BETWEEN-FAILURE (MTBF)- a basic measure of reliability for repairable items: The mean number of life units during which all parts of the item perform within their specified limits, during a particular measurement interval under stated conditions. (MIL-STD-721-C)

MEAN-TIME-BETWEEN-MAINTENANCE (MTBM)- a measure of the reliability taking into account maintenance policy. The total number of life units expended by a given time, divided by the total number of maintenance events (scheduled and unscheduled) due to that item. (MIL-STD-721-C)

MEAN-TIME-TO-REPAIR (MTTR)- a basic measure of maintainability: The sum of corrective maintenance times at any specific level of repair, divided by the total number of failures within an item repaired at that level, during a particular interval under stated conditions. (MIL-STD-721-C)

MONTE CARLO MODEL- a method for solving non-probabilistic problems using parameters which satisfy the requirements of the original problem. In this method, the data of interest are plotted or tabulated as a cumulative probability distribution function, i.e. the probability P of x, or more, or of x or less "successes" against x on the horizontal axis. (Hines & Montgomery, 1972)

NORMAL DISTRIBUTION- a frequency distribution that is characterized by a bell-shaped and symmetrical pattern. (Hines & Montgomery, 1972)

OPERATIONAL R & M VALUE- a measure of reliability or maintainability that includes the combined effect of item design, installation, quality, environment, operation, maintenance, and repair. (MIL-STD-721-C)

RELIABILITY- (1) the duration or probability of failure-free performance under stated conditions; (2) the probability that an item can perform its intended function for a specified interval under stated conditions. (MIL-STD-721-C)

SKEW- a measure of the deviation from symmetry about the point of central tendency. [A normal distribution has zero skewness.] It is the third moment about the mean. (Hines & Montgomery, 1972)

SYSTEM- a composite of equipment and skills, and techniques capable of performing or supporting an operational role, or both. A complete system includes all equipment, related facilities, material. software, services, and personnel required for its operation and support to the degree that it can be considered self-sufficient in its intended operational environment. (MIL-STD-721-C)

SYSTEM R & M PARAMETER- a measure of reliability or maintainability in which the units of measurement are directly related to operational readiness, mission success, maintenance manpower cost, or to logistic support cost. (MIL-STD-721-C)

TIME- the universal measure of duration. The general word "time" will be modified by an additional term when used in reference to operating time, mission time, test time,etc. In general, when used in expressions such as "Mean-Time-Between-Failure," time stands for "life units" which must be specifically defined whenever the general term refers to a particular item. (MIL-STD-721-C)

UPTIME RATIO- a composite measure of operational availability and dependability that includes the combined effects of item design, installation, quality, environment, operation, maintenance, repair and logistic support: The quotient of uptime divided by uptime plus downtime. (MIL-STD-721-C)

WEAROUT- the process which results in an increase of the failure rate or probability of failure with increasing number of life units. (MIL-STD-721-C)

WEIBULL DISTRIBUTION- a family of density functions widely used to describe failure rate characteristics in reliability analysis. The descriptions are in the form of probability density curves. (Hines & Montgomery, 1972)

REFERENCES

General

1. A.J. Duncan, *Quality Control and Industrial Statistics*, 4th ed., Richard D. Irwin, Inc., Homewood, IL, 1974.

2. W.W. Hines & D.C. Montgomery, *Probability and Statistics in Engineering and Management Science*, John Wiley and Sons, New York, 1972.

3. Herbert Schildt, *Advanced Turbo Prolog*™, Osborne McGraw-Hill, Berkeley, 1987.

4. *Turbo Prolog: Owner's Manual*, Borland International, Inc., Scotts Valley, CA, 1986.

Military Documents

5. MIL-HDBK-266, *Application of Reliability-Centered Maintenance to Naval Aircraft, Weapon Systems and Support Equipment.*

6. MIL-STD-721-C, *Definitions of Terms for Reliability and Maintainability.*

7. MIL-STD-1388-1A, *Logistic Support Analysis.*

APPENDIX A

R710- FLIGHT ACTIVITY AND INVENTORY REPORT

[BLANK PAGE]

R710 - FLIGHT ACTIVITY AND INVENTORY REPORT

Introduction

The following sample illustrates the format of the R710 Report and the data file that is created from the R710 Report. The data file, Sqrdqt.sqr, is the output file created from Step 13 of the NAIL System. It is a simplified data set created from the R710 Report.

[BLANK PAGE]

Sqrdqr.sqr
Output from Step 13. Fix R710
Input file - R710.DAT

ORG :AB3

A	B	C	D	E	F	G	H	I
365	7.4	144	74	5393	57	34	71	89
395	6.5	199	93	4700			1	2
426	5.2	284	131	3748				1
456	8.4	525	285	6097				3
487	7.0	389	171	5069				
517	10.1	376	238	7285	89	42	45	78
548	10.4	399	205	7551	341	168	155	214

A: Date.
B: Total Hours in Service Divided by 720.
C: Total Flight Hours Report per Month.
D: Total Number of Flights Reported per Month.
E: Total Hours in Readiness Reporting Status per Month.
F: Total Flight Hours Involving Ship Operations per Month.
G: Total Number of Flights Involving Ship Operations per Month.
H: Total Number of Arrested Landings per Month.
I: Total Number of Catapult-Assisted Take-offs per Month.

REPORT NBR: R0710
PREPARED: 01 NOV 1986

EQUIPMENT CONDITION ANALYSIS

PAGE 1

FLIGHT ACTIVITY AND INVENTORY REPORT
REPORT PERIOD: DEC 84 THRU JUN 85

URG 1AB3

DATE	AVERAGE NUMBER REPORTING INVENTORY	FLIGHT HOURS	NUMBER SORTIES	HOURS EQUIPMENT IN SERVICE	---- SHIP OPS --- HOURS	FLIGHTS	ARREST LANDINGS	CAT SHOTS
DEC 84	7.4	144	74	5393	57	34	71	89
JAN 85	6.5	199	93	4700			1	2
FEB 85	5.2	284	131	3748				1
MAR 85	8.4	525	285	6097				3
APR 85	7.0	389	171	5069				
MAY 85	10.1	376	238	7285	89	42	45	78
JUN 85	10.4	399	205	7551	341	168	155	214
-AC TOTALS	7.9	2316	1197	39843	487	244	272	387

REPORT NBR: R0710
PREPARED: 01 NOV 1986

EQUIPMENT CONDITION ANALYSIS

PAGE 1

FLIGHT ACTIVITY AND INVENTORY REPORT
REPORT PERIOD: DEC 84 THRU JUN 85

URG 1AB3

DATE	AVERAGE NUMBER REPORTING INVENTORY	FLIGHT HOURS	NUMBER SORTIES	HOURS EQUIPMENT IN SERVICE	---- SHIP OPS --- HOURS	FLIGHTS	ARREST LANDINGS	CAT SHOTS
xxx xx	xxxxx	xxxxxx	xxxxxx	xxxxxxxx	xxxxxxxx	xxxxxx	xxxx	xxxx
A	B	C	D	E	F	G	H	I

A: Date MMM YY.
B: Total Hours in Service Divided by 720.
C: Total Flight Hours Report per Month.
D: Total Number of Flights Reported per Month.
E: Total Hours in Readiness Reporting Status per Month.
F: Total Flight Hours Involving Ship Operations per Month.
G: Total Number of Flights Involving Ship Operations per Month.
H: Total Number of Arrested Landings per Month.
I: Total Number of Catapult-Assisted Take-offs per Month.

APPENDIX B

R520- INDIVIDUAL MAINTENANCE ACTION REPORT

[BLANK PAGE]

R520 - INDIVIDUAL MAINTENANCE ACTION REPORT

Introduction

The following sample illustrates the format of the R520 Report and the data files created from this report. Step 1 of the NAIL System strips the headings off the R520 Report and converts the dates to sequential numbers. The output file from Step 1 will have a file extension of .DTA. This file is used by Step 2 of the NAIL System to calculate the time between failures. Step 2 will create an output file with the file extension of .DTB. Step 3 includes the data from the R710 data set with the data set created by Step 2. The combined data set will have the file extension of .DTC. Step 19 uses this file to create a data set for input into the Regression Analysis Procedure.

[BLANK PAGE]

R520A6.DTD
Output from Step 19. Format Data for Regression Analysis.

```
 5          (* Number of Variables *)
     8      (* MTBF   *)
   338      (* Variable 1 - Job Control Number - JCN Date
    17      (* Variable 2 - Total Hours in Service *)
   380      (* Variable 3 - Flight Hours *)
   175      (* Variable 4 - Number Flights *)
 12536      (* Variable 5 - Hours in Readiness *)
     8
   375
     5
    59
    31
  3543
     9
   375
     5
    59
    31
  3543
```

R520A6.DTB
Output from Step 2. Evaluate Time Between Failures.
Input file - R520A6.DTA

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
152628	1113000	128B102023	26512	0	PJ5	338	578		9	339		R	B	R	7		170	170		A4751			E 1	0
161247	1113000	1128B403788170	26512	0	P66	375	805		487	503		H	B	R	7		170	846		A1713			E 1	0
%161247	1113000	1128B4037817	26512	0	P66	375	805		487	503										A1713			E 1	0
161247	1113000	128BH100287	26512	0	P66	375	806		486	504		H	B	R	C		170	780		A1713			E 1	0
149937	1113000	128B102015	26512	222301	AE4	403	046	B	417	416		L	D	R	C		170	520		A2942			E 1	0
151596	1113000	128EC120125	26512	0	KB8	424	449		429	425		H	B	R	C		381	381		A3773			E 1	0
159312	1113000	128B113753	26512	A6754	WA5	435	021		436	437		Y	B	R	C		190	070		A2404			E 1	0
%159312	1113000	128B113753	26512	0000	WA5	435	021		436	437										A2404			E 1	0
159192	1113000	128B102085	26512	000	AG5	457	072		458			H	B	R			170			A0123			E 1	0
%159192	1113000	128B102085	26512	0	AG5	457	072		458	457					7			170		A3861			E 2	0
151579	1113000	128B10213627	26512	0000	AB4	469	071		470	496		L	D	R	C		170	070		A5462			E 1	0
%151579	1113000	12B10213627	26512	0000	AB4	469	071		470	496										A5462			E 1	0
159905	1113000	128BH100271	26512	0838	FA7	507	086		508	507		H	B	R	7		070	070		A3312			E 1	0
%159905	1113000	128BH100271	26512	000	FA7	507	086		508	507										A0000			E 1	0
155662	1113000	128B102085	26512	0	P22	507	105		534	537		Q	S	R	1		780	242		A3171			E 1	0
159178	1113000	128WH100131	26512	0	AC3	523	246		545	542		R	B	R	7		170	846		A3858			E 1	0
159178	1113000	128B11375602	26512	0	AC3	529	022		530	529		H	B	R	7		846	780		A3829			E 1	0
%159178	1113000	128B11375602	26512	000	AC3	529	022		530	529										A3829			E 1	0

A: Bureau Number.
B: Work Unit Code.
C: Part Number.
D: Manufacture Code.
E: Serial Number.
F: Job Control Number. (Organization)
G: Job Control Number. (JCN Date)
H: Job Control Number. (Sequence)
I: Job Control Number. (JCN Suffix)
J: Organization Level Action Date.
K: Intermediate Level Action Date.
L: Depot Level Action Date.
M: When Discovered Code.
N: Type Maintence Code.
O: Action Taken Code. (Organization Level)
P: Action Taken Code. (Intermediate Level)
Q: Action Taken Code. (Depot Level)
R: Organization Level.
S: Intermediate Level Malfunction Code.
T: Depot Level Malfunction Code.
U: Meter Block 1 Reading.
V: Meter Block 2 Reading.
W: Meter Block 3 Reading.
X: Card Code.
Y: Time Between Failures

R520A6.DTC
Output from Step 3. Include 710 Data.
Input files - R520A6.DTB, Sqrdqt.sqr

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A.a	B.b
152628	1113000	128B102023	26512	0	PJ5	338	578		9	339	R	B	R	7			170	170		A4751	E	1	0	8	17	380	175	12536
161247	1113000	1128B4037888170	26512	0	P66	375	805		487	503	H	B	R	7			170	846		A1713	E	1	0	8	5	59	31	3543
&161247	1113000	1128B4037817	26512	0	P66	375	805		487	503										A1713	E	1	0	9	5	59	31	3543
161247	1113000	128BH100287	26512	0	P66	375	806		486	504	H	B	R	C			170	780		A1713	E	1	0	8	5	59	31	3543
149937	1113000	128B102015	26512	222301	AE4	403	046	B	417	416	L	D	R	C			170	520		A2942	E	1	0	10	13	108	53	9515
151596	1113000	128EC120125	26512	0	KBB	424	449		429	425	H	B	R	C			381	381		A3773	E	1	0	10	4	69	32	2976
159312	1113000	128B113753	26512	A6754	WA5	435	021		436	437	Y	B	R	C			190	070		A2404	E	1	0	11	3	38	28	2090
&159312	1113000	128B113753	26512	0000	WA5	435	021		436	437										A2404	E	1	0	10	3	38	28	2090
159192	1113000	128B102085	26512	000	A65	457	072		458		H	B	R				170			A0123	E	1	0	10	13	485	218	9501
&159192	1113000	128B102085	26512	0	A65	457	072		458	457					7			170		A3861	E	2	0	10	13	485	218	9501
151579	1113000	128B10213627	26512	0000	AB4	469	071		470	496	L	D	R	C			170	070		A5462	E	1	0	10	16	530	233	11410
&151579	1113000	12810213627	26512	0000	AB4	469	071		470	496										A5462	E	1	0	10	16	530	233	11410
159905	1113000	128BH100271	26512	0B38	FA7	507	086		508	507	H	B	R	7			070	070		A3312	E	1	0	11	8	352	172	5525
&159905	1113000	128BH100271	26512	000	FA7	507	086		508	507										A0000	E	1	0	11	8	352	172	5525
155662	1113000	128B102085	26512	0	P22	507	105		534	537	Q	S	R	1			780	242		A3171	E	1	0	11	3	75	44	2119
159178	1113000	128BH100131	26512	0	AC3	523	246		545	542	R	B	R	7			170	846		A3858	E	1	0	11	16	404	230	11672
159178	1113000	128B11375602	26512	0	AC3	529	022		530	529	H	B	R	7			846	780		A3829	E	1	0	11	16	404	230	11672
&159178	1113000	128B11375602	26512	000	AC3	529	022		530	529										A3829	E	1	0	11	16	404	230	11672

A: Bureau Number.
B: Work Unit Code.
C: Part Number.
D: Manufacture Code.
E: Serial Number.
F: Job Control Number. (Organization)
G: Job Control Number. (JCN Date)
H: Job Control Number. (Sequence)
I: Job Control Number. (JCN Suffix)
J: Organization Level Action Date.
K: Intermediate Level Action Date.
L: Depot Level Action Date.
M: When Discovered Code.
N: Type Maintence Code.
O: Action Taken Code. (Organization Level)
P: Action Taken Code. (Intermediate Level)
Q: Action Taken Code. (Depot Level)
R: Organization Level.
S: Intermediate Level Malfinction Code.
T: Depot Level Level Malfunction Level.
U: Meter Block 1 Reading.
V: Card Code.
W: Time Between Failures.
X: MTBF.
Y: Total Hours in Service.
Z: Flight Hours.
A.a: Number Flights.
B.b: Hours in Readiness.

R520A6.DTA
Output from Step 1. Remove Report Headings.
Input file - R520A6.DAT

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
152628	1113000	128B102023	26512	0	PJ5	338	578		9	339		R	B	R	7		170	170		A4751			E 1
161247	1113000	1128B403788170	26512	0	P66	375	805		487	503		H	B	R	7		170	846		A1713			E 1
&161247	1113000	1128B4037817	26512	0	P66	375	805		487	503										A1713			E 1
161247	1113000	128BH100287	26512	0	P66	375	806		486	504		H	B	R	C		170	780		A1713			E 1
149937	1113000	128B102015	26512	222301	AE4	403	046	B	417	416		L	D	R	C		170	520		A2942			E 1
151596	1113000	128EC120125	26512	0	KBB	424	449		429	425		H	B	R	C		381	381		A3773			E 1
159312	1113000	128B113753	26512	A6754	WA5	435	021		436	437		Y	B	R	C		190	070		A2404			E 1
&159312	1113000	128B113753	26512	0000	WA5	435	021		436	437										A2404			E 1
159192	1113000	128B102085	26512	000	AG5	457	072		458			H	B	R			170			A0123			E 1
&159192	1113000	128B102085	26512	0	AG5	457	072		458	457					7			170		A3861			E 2
151579	1113000	128B10213627	26512	0000	AB4	469	071		470	496		L	D	R	C		170	070		A5462			E 1
&151579	1113000	12810213627	26512	0000	AB4	469	071		470	496										A5462			E 1
159905	1113000	128BH100271	26512	0838	FA7	507	086		508	507		H	B	R	7		070	070		A3312			E 1
&159905	1113000	128BH100271	26512	000	FA7	507	086		508	507										A0000			E 1
155662	1113000	128B102085	26512	0	P22	507	105		534	537		Q	S	R	1		780	242		A3171			E 1
159178	1113000	128WH100131	26512	0	AC3	523	246		545	542		R	B	R	7		170	846		A3858			E 1
159178	1113000	128B11375602	26512	0	AC3	529	022		530	529		H	B	R	7		846	780		A3829			E 1
&159178	1113000	128B11375602	26512	000	AC3	529	022		530	529										A3829			E 1

A: Bureau Number.
B: Work Unit Code.
C: Part Number.
D: Manufacture Code.
E: Serial Number.
F: Job Control Number. (Organization)
G: Job Control Number. (JCN Date)
H: Job Control Number. (Sequence)
I: Job Control Number. (JCN Suffix)
J: Organization Level Action Date.
K: Intermediate Level Action Date.
L: Depot Level Action Date.
M: When Discovered Code.
N: Type Maintence Code.
O: Action Taken Code. (Organization Level)
P: Action Taken Code. (Intermediate Level)
Q: Action Taken Code. (Depot Level)
R: Organization Level.
S: Intermediate Level Malfunction Code.
T: Depot Level Malfunction Code.
U: Meter Block 1 Reading.
V: Meter Block 2 Reading.
W: Meter Block 3 Reading.
X: Card Code.

REPORT NBR: R0520 EQUIPMENT CONDITION ANALYSIS PAGE 1

PREPARED: 03 NOV 1986

INDIVIDUAL MAINTENANCE ACTION RECORD

REPORT PERIOD: DEC 84 THRU JUN 85

I/M: A6

---------- EQUIPMENT DESCRIPTION ----------

TYPE EQPT CODE	BUREAU NUMBER	WORK UNIT CODE	PI	PART NBR	MFG CODE	SERIAL NBR	JCN ORG	JCN DATE	JCN SEQ	JCN SUF	ACTION DATE ORG	ACTION DATE IMA	ACTION DATE DEP	W D	T M	ACT TAKEN D O	ACT TAKEN I	ACT TAKEN D	MALFUNCTION CODES ORG	MALFUNCTION CODES IMA	MALFUNCTION CODES DEP	METER TIMES 1	METER TIMES 2	METER TIMES 3	C O	M L
xxxx	xxxxxx	xxxxxxx	xx	xxxxxxxxxxxxx	xxxxx	xxxxxxxxxx	xxx	xxxxx	xxx	xx	xxxx	xxxx	xxxx	x	x	x	x	x	xxx	xxx	xxx	xxxxx	xxxxx	xxxxx	x	
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	

A: Type Equipment Code.
B: Bureau Number.
C: Work Unit Code.
D: Position Indicator.
E: Part Number.
F: Manufacture Code.
G: Serial Number.
H: Job Control Number. (Organization)
I: Job Control Number. (JCN Date)
J: Job Control Number. (Sequence)
K: Job Control Number. (JCN Suffix)
L: Organization Level Action Date.
M: Intermediate Level Action Date.
N: Depot Level Action Date.
O: When discovered Code
P: Type Maintence Code.
Q: Action Taken Code. (Organization Level)
R: Action Taken Code. (Intermediate Level)
S: Action Taken Code. (Depot Level)
T: Organization Level.
U: Intermediate Level Malfunction Code.
V: Depot Level Malfunction Code.
W: Meter Block 1 Reading.
X: Meter Block 2 Reading.
Y: Meter Block 3 Reading.
Z: Card Code.

REPORT NBR: R0520 EQUIPMENT CONDITION ANALYSIS PAGE 1

PREPARED: 03 NOV 1986

INDIVIDUAL MAINTENANCE ACTION RECORD

REPORT PERIOD: DEC 84 THRU JUN 85

W/M: A6

---------- EQUIPMENT DESCRIPTION ----------

TYPE EQPT CODE	BUREAU NUMBER	WORK UNIT CODE	PI PART NBR	MFG CODE	SERIAL NBR	------ JCN ------ ORG	DATE	SED	SUF	ACTION DATE ORG	IMA	DEP	W D	T M	ACT TAKEN O	I	D	MALFUNCTION CODES ORG	IMA	DEP	-- METER TIMES -- 1	2	3	C	L M L
AAEF	152628	1113000	128B102023	26512	0	PJ5	84338	578		5009	4339		R	B	R	7		170	170		A4751			E	1
			128B102023	26512	0																A4751			G	
AAED	161247	1113000	1128B4037B8170	26512	0	P66	85010	805		5122	5138		H	B	R	7		170	846		A1713			E	1
			1128B4037817	26512	0																A1713			E	1
			1128B40377817	26512	0																A1713			G	
AAED	161247	1113000	128BH100287	26512	0	F66	85010	806		5121	5139		H	B	R	C		170	780		A1713			E	1
			128BH100287	26512	0																A1713			G	
AAEF	149937	1113000	128B102015	26512	222301	AE4	85038	046	B	5052	5051		L	D	R	C		170	520		A2942			E	1
			128B102015	26512	222301																A2942			G	
AAEB	151596	1113000	128EC120125	26512	0	KBB	85059	449		5064	5060		H	B	R	C		381	381		A3773			E	1
			128EC120125	26512	0																A3773			G	
AAEG	159312	1113000	128B113753	26512	A6754	WA5	85070	021		5071	5072		Y	B	R	C		190	070		A2404			E	1
			128B113753	26512	0000																A2404			E	1
			128B11375601	26512	0000																A2404			G	
AAEG	159192	1113000	128B102085	26512	000	AB5	85092	072		5093			H	B	R			170			A0123			E	1
			128B102085	26512	000																A0123			G	
			128B102085	26512	0						5092					7			170		A3861			E	2
AAEF	151579	1113000	128B10213627	26512	0000	AB4	85104	071		5105	5131		L	D	R	C		170	070		A5462			E	1
			12810213627	26512	0000																A5462			E	1
			12810213627	26512	0000																A5462			G	
AAEG	159905	1113000	128BH100271	26512	0838	FA7	85142	086		5143	5142		H	B	R	7		070	070		A3312			E	1
			128BH100271	26512	000																A0000			E	1
			128BH100271	26512	A65936																A0000			G	
AAEG	155662	1113000	128B102085	26512	0	P22	85142	105		5169	5172		Q	S	R	1		780	242		A3171			E	1
			128B102085	26512	0																A3171			G	
AAEG	159178	1113000	128WH100131	26512	0	AC3	85158	246		5180	5177		R	B	R	7		170	846		A3858			E	1
			128WH100131	26512	0																A3859			G	
AAEG	159178	1113000	128B11375602	26512	0	AC3	85164	022		5165	5164		H	B	R	7		846	780		A3829			E	1
			128B11375602	26512	000																A3829			E	1
			128B11375602	26512	A6791																A3829			G	

APPENDIX C

REPLACEMENT COMPONENT METHODOLOGY

[BLANK PAGE]

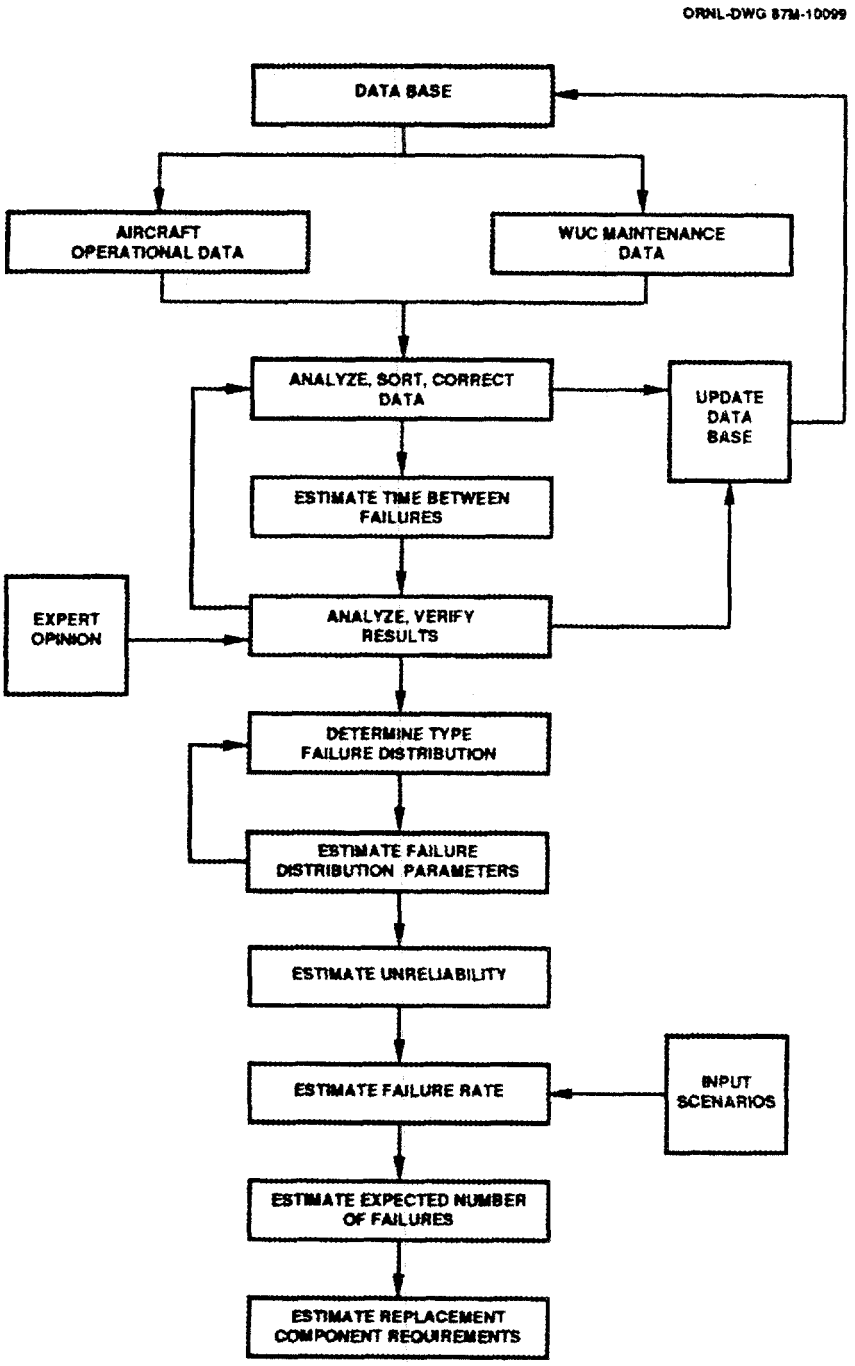
REPLACEMENT COMPONENT METHODOLOGY

The following chart illustrates the procedure used to estimate component failure and replacement requirements on selected WUCs. The initial step in the process is to analyze, sort, and correct the data. The analysis, sorting, and correcting process is an iterative process in which the time between failures is estimated and then analyzed to verify that the results are reasonable. This verification task frequently requires that data be screened carefully and corrected, where possible. After the iterative process is complete, TBF is estimated by regression analysis.

The regression analysis of selected WUCs identifies independent variables (e.g., time, flight-hours, landings, and takeoff) that determine TBF. After the regression is completed, failure distribution is estimated by examining the frequency distribution of the individual parameters (TBF, flight-hours, calendar time). This frequency distribution determines the type of distribution (e.g., exponential, Normal, Weibull, or other standard reliability distributions). Given the type of failure distribution, the probability density function parameters (e.g., sigma, mu, beta, or lambda) are estimated. Once this type of failure distribution is established, unreliability may be estimated. From this estimate, the appropriate failure rate for the component is established. With an estimate of the failure rate, the expected number of flight-hours may be projected.

The last step in this analysis is to estimate the replacement components required for the number of failures projected. This step requires significant depot-level data identifying parts that were either repaired or replaced. From this depot-level data, the number of failures which resulted in component replacement may be determined.

[BLANK PAGE]



REPLACEMENT COMPONENT METHODOLOGY

[BLANK PAGE]

APPENDIX D

REGRESSION ANALYSIS

[BLANK PAGE]

REGRESSION ANALYSIS

Introduction

The number of repairs and replacements for a part j stored in a supply depot can be thought of as a function, F(j,r), of n number of variables, where the parameters r is a vector representing the set of variables (V_1, V_2,...,V_n). Clearly, if historical data are available on repairs and replacements, if a sufficient number of V's can be identified, and if the historical data can be related to the various variables in a meaningful way; then, an empirical representation of the repair and replacement for a single part can be constructed. The empirical equations can be installed in computer programs that will provide the needed forecasting tools.

It is assumed that a function relation, denoted by f(r), exists between the repair and replacement of a part and one or more variables. The first variable V_1 in a list of candidates could be a challenge rate, e.g. flight hours, which would tend to influence repair and replacement rates for all parts of a weapon system. The other variables are observable and predictable events that can be used to evaluate f(r).

In order to develop the f(r) function, the assumption of variable independence is made, i.e. that V_1 is independent of V_2. It is also assumed that the replacement and repair functions will be linear additive functions of the independent variables (V's). Thus, the model is given by

$$f = b_0 + b_1V_1 + b_2V_2 \ldots + b_nV_n \qquad (1)$$

where b_n are the coefficients determined using statistical methods. These methods minimize the standard error between historical data and function predictions given by

$$SD = \text{sqrt}(\text{sum } [f(ri) - fi]/[N - n] \qquad (2)$$

where fi is an actual data point that corresponds to the function f(ri). The vector ri indicates the values of Vi that are assumed to correlate with the data point(s). N is the total number of points and n is the number of variables used for the evaluation. Equation (2) is used to evaluate functional assumptions and test the significance of proposed V's. The first cut at minimizing SD uses least squares analysis of one or more V's, e.g. system performance under stable and mobilization operation. The tests have used blind projection to test functional assumptions.

Obviously, this approach need not find all V's to reduce projection error; the identification of one variable will reduce projection error. The number of V's identified will simply increase the capability of the model to handle various scenarios and increase the confidence level between forecasts and actual requirements.

In order to compare the effect of adding new variables to the prediction equation, the coefficient of correlation must be evaluated for each variable used in the equation. This will help identify variables that are themselves functions of one

another (e.g. carrier landings and hours of operation at sea). The coefficient of determination is used to compare or explain the total variation which is not explained by the regression line, or simply the variance of the function divided by the variance of the average subtracted from one. The square root of this value is the coefficient of correlation. This parameter has a value of one if all the data points fall on the regression line. This value implies perfect correlation between the independent and dependent variables. Conversely, its value is zero when no correlation is present.

Future versions of the program that use the coefficient matrix generated by this study will evaluate the spare part requirements under various input conditions. The output will be based on two types of requirements: calculations based on input probability of shortage, or number, of spares; calculations based on the probability of shortage. The first code will be a direct I/O calculation and the second will involve stoachistic evaluation of a scenario. This latter modeling, a Monte Carlo analysis, requires greater computer run times, but it is a very powerful management tool for understanding probability distributions.

* REGRESSION ANALYSIS *

THESE ARE THE GENERAL COMMENTS ENTERED WHEN FIRST ENTERING THE NAIL SYSTEM. THE COMMENTS ARE PRINTED ON THE FIRST PAGE OF ALL REPORTS. THE REPORTS GENERATED DURING THIS SESSION ARE ON AIRCRAFT - A6, WUC -1113000.

Data File used for study: 1R520A6.DTD
Date of study: 02/11/87
Time of study: 11:34:14
Number of X/Y Data (Top): 24
Number of Variables: 5

[BLANK PAGE]

* Regression Analysis *

Data File used for study: 1R520A6.DTD
Date of study: 02/11/87

No.	F	f12	df12	f1	df1	V1	V2	...		
1	8.0	8	0	8	-0.3	338.0	17.0	380.0	1751.0	2536.0
2	8.0	9	-1	9	-0.8	375.0	5.0	59.0	31.0	3543.0
3	9.0	9	0	9	0.2	375.0	5.0	59.0	31.0	3543.0
4	8.0	9	-1	9	-0.8	375.0	5.0	59.0	31.0	3543.0
5	10.0	9	1	9	0.8	403.0	13.0	108.0	53.0	9515.0
6	10.0	10	0	10	0.5	424.0	4.0	69.0	32.0	2976.0
7	11.0	10	1	10	1.3	435.0	3.0	38.0	28.0	2090.0
8	10.0	10	0	10	0.3	435.0	3.0	38.0	28.0	2090.0
9	10.0	10	0	10	-0.0	457.0	13.0	485.0	218.0	9501.0
10	10.0	10	0	10	-0.0	457.0	13.0	485.0	218.0	9501.0
11	10.0	10	-0	10	-0.2	469.0	16.0	530.0	2331.0	1410.0
12	10.0	10	-0	10	-0.2	469.0	16.0	530.0	2331.0	1410.0
13	11.0	11	0	11	0.2	507.0	8.0	352.0	172.0	5525.0
14	11.0	11	0	11	0.2	507.0	8.0	352.0	172.0	5525.0
15	11.0	11	-0	11	0.2	507.0	3.0	75.0	44.0	2119.0
16	11.0	11	0	11	0.0	523.0	16.0	404.0	2301.0	1672.0
17	11.0	11	0	11	-0.1	529.0	16.0	404.0	2301.0	1672.0
18	11.0	11	0	11	-0.1	529.0	16.0	404.0	2301.0	1672.0
19	11.0	11	-0	11	-0.1	534.0	11.0	480.0	222.0	7773.0
20	11.0	11	-0	11	-0.1	534.0	11.0	480.0	222.0	7773.0
21	11.0	11	-0	11	-0.2	536.0	16.0	530.0	2331.0	1410.0
22	11.0	11	-0	11	-0.2	536.0	16.0	530.0	2331.0	1410.0
23	11.0	11	-0	11	-0.2	541.0	13.0	108.0	53.0	9515.0
24	11.0	11	-0	11	-0.2	541.0	13.0	108.0	53.0	9515.0
25	10.2	10	0	10	0.4	472.3	10.8	294.5	816.1	4468.3

```
!   f1 = a + b*V1
!    a =   3.3916644944E+00
!    b =   1.4520117514E-02
!=====================================================
!  f12 = C1 + C2*V1 + C3*V2 + ... + Cn*Vn
!  C 1 =   3.3023217777E+00
!  C 2 =   1.5417320399E-02
!  C 3 =  -5.5997432442E-02
!  C 4 =  -2.8442564735E-04
!  C 5 =   2.0691984901E-04
!  C 6 =   4.1870676341E-05
   STANDARD DEVIATION OF DATA =   1.0320936931E+00
   COEF. OF CORRELATION (f12) =   9.1424253693E-01
   COEF. OF CORRELATION ( f1) =   9.0239231850E-01
   Delta                      =   1.1850218437E-02
```

[BLANK PAGE]

APPENDIX E

MASTER INDEX OF REPAIRABLES (MIR)

[BLANK PAGE]

MASTER INDEX OF REPAIRABLES

Introduction

The MIR report for the number of projected operations occurring at the flight line (operational), intermediate, and depot level over a five-year period is illustrated in this appendix. A WUC item is selected along with a time interval and confidence interval. The number of failures, repairs, and/or replacements at each level is projected for the desired confidence interval. This interval is interpreted in the following example as 5% probability that actual events will exceed the projected values.

[BLANK PAGE]

* MASTER INDEX OF REPAIRABLES (MIR) *

THESE ARE THE GENERAL COMMENTS ENTERED WHEN FIRST ENTERING THE NAIL SYSTEM. THE COMMENTS ARE PRINTED ON THE FIRST PAGE OF ALL REPORTS. THE REPORTS GENERATED DURING THIS SESSION ARE ON AIRCRAFT - A6, WUC -1113000.

Data File used for study:	1R520A6.DTB
Date of study:	02/11/87
Time of study:	11:26:32
Confidence level (%):	95 %

[BLANK PAGE]

Number of Repairs/Removals

Date	Obs.	fit	del.
DEC. 84	1	1	0
JAN. 85	3	2	1
FEB. 85	2	3	-1
MAR. 85	2	3	-1
APR. 85	4	4	-0
MAY 85	3	5	-2
JUNE 85	9	6	3
	====	====	====
Adv.	3	3	2

```
Max Time in data        =   541   JUNE  85
Min time in data        =   338   DEC.  84
Total number of data    =    24
Number to be cal.       =    24
Mean time between fail. =   203.000
Mean value for unseen   =   203.000
Max time for cal. time  =   304.500
---------------------------------
Number of repairs/removals (O-level) =  15
Number of repairs/removals (I-level) =   6
Assumed  repairs/removals (D-level) =   9
Observed repairs/removals (D-level) =   0
* No data on D - level so best aprox. made. *
```

* Master Index of Repariables (MIR) *

WUC: 1113000

Year	Operational	Intermediate	Depot
1	93	37	56
2	177	70	106
3	260	104	156
4	344	137	206
5	427	171	256
Total	1301	519	780
Adv.	260	104	156

[BLANK PAGE]

APPENDIX F

MONTHLY FAILURE RATES

[BLANK PAGE]

* Monthly Failure Rates *

THESE ARE THE GENERAL COMMENTS ENTERED WHEN FIRST ENTERING THE NAIL SYSTEM. THE COMMENTS ARE PRINTED ON THE FIRST PAGE OF ALL REPORT. THE REPORTS GENERATED DURING THIS SESSION ARE ON AIRCRAFT - A6, WCU -1113000.

Data File used for study: 1R520A6.DTB
Date of study: 02/11/87
Time of study: 11:33:02

[BLANK PAGE]

Sqrd 1 = AB3	Sqrd 13 = FA5	Sqrd 25 = PF4	Sqrd 37 = P67
Sqrd 2 = AB4	Sqrd 14 = FA7	Sqrd 26 = PH5	Sqrd 38 = P6B
Sqrd 3 = AC3	Sqrd 15 = FA9	Sqrd 27 = PJ5	Sqrd 39 = QH0
Sqrd 4 = AD4	Sqrd 16 = FB4	Sqrd 28 = P22	Sqrd 40 = SS0
Sqrd 5 = AE4	Sqrd 17 = GF7	Sqrd 29 = P6A	Sqrd 41 = WAM
Sqrd 6 = AF5	Sqrd 18 = GGB	Sqrd 30 = P6B	Sqrd 42 = WAU
Sqrd 7 = AG5	Sqrd 19 = KA9	Sqrd 31 = P61	Sqrd 43 = WA5
Sqrd 8 = AW3	Sqrd 20 = KBB	Sqrd 32 = P62	Sqrd 44 = WNF
Sqrd 9 = BB0	Sqrd 21 = PB3	Sqrd 33 = P63	Sqrd 45 =
Sqrd 10 = FAF	Sqrd 22 = PC5	Sqrd 34 = P64	Sqrd 46 =
Sqrd 11 = FAG	Sqrd 23 = PD5	Sqrd 35 = P65	Sqrd 47 =
Sqrd 12 = FA3	Sqrd 24 = PEB	Sqrd 36 = P66	Sqrd 48 =

Number of aircraft as a function of time & sqn. Sheet 1 of 2

Date	Sqr. Number 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
DEC. 84	7.4	15.8	16.2	20.3	13.2	11.8	13.1	12.6	4.1	4.1	---	9.5	9.3	8.2	9.5	10.7	7.3	7.9	3.5	4.1	16.5	13.8	12.3	17.2
JAN. 85	6.5	14.5	13.7	18.1	9.2	8.5	13.4	11.6	4.1	4.1	---	5.8	9.2	6.2	6.9	7.2	3.2	5.7	2.8	4.1	16.5	13.1	13.2	12.9
FEB. 85	5.2	15.4	12.1	15.8	7.3	9.2	11.8	11.8	3.7	3.7	---	5.8	8.4	6.8	7.6	6.8	4.7	4.6	3.2	3.7	14.9	9.9	12.7	12.8
MAR. 85	8.4	17.7	13.2	17.5	8.1	11.2	8.1	14.4	4.2	4.1	---	9.4	9.3	8.3	7.6	7.8	6.1	5.0	4.0	4.1	15.5	8.6	11.4	14.8
APR. 85	7.0	16.8	14.4	16.5	8.5	11.0	6.9	11.8	5.0	3.3	0.6	9.8	8.8	8.2	5.9	8.2	4.8	4.7	3.9	5.0	15.0	10.9	11.1	16.1
MAY 85	10.1	17.4	13.7	15.1	8.9	9.8	6.0	13.5	5.1	---	4.1	3.0	9.5	9.1	5.9	8.5	4.9	4.9	4.1	4.7	14.4	8.8	13.8	13.5
JUNE 85	10.4	16.6	15.5	10.3	7.6	9.4	7.2	13.4	5.0	---	4.0	6.7	9.0	8.3	6.5	8.9	6.5	6.2	3.8	4.0	6.9	7.3	13.4	13.5
JULY 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
AUG. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
SEP. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
OCT. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
NOV. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
DEC. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JAN. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
FEB. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
MAR. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
APR. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
MAY 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JUNE 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JULY 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
AUG. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
SEP. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
OCT. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
NOV. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
DEC. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JAN. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
FEB. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
MAR. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
APR. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
MAY 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JUNE 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JULY 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
AUG. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
SEP. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
OCT. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
NOV. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
DEC. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JAN. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
FEB. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
MAR. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
APR. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
MAY 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JUNE 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JULY 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
AUG. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
SEP. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
OCT. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
NOV. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Number of aircraft as a function of time & sqd. Sheet 2 of 2

Date	:-------- Sqr. Number --------)																				
	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	Tot.
DEC. 84	9.9	13.4	17.4	3.4	2.9	4.9	4.1	3.7	---	4.1	2.3	4.9	2.0	3.4	11.2	4.9	4.1	4.0	3.1	54.1	406.2
JAN. 85	8.2	15.4	16.0	2.0	2.7	4.1	4.1	4.1	4.1	3.8	1.4	5.1	2.3	3.9	10.2	4.8	4.1	4.0	2.6	15.1	328.5
FEB. 85	6.8	11.6	12.1	2.6	2.6	3.7	3.7	3.7	3.1	3.6	0.9	3.3	3.3	2.4	7.8	3.7	---	2.8	2.3	84.4	362.3
MAR. 85	6.8	10.4	13.6	3.1	3.3	4.1	2.8	4.1	1.3	4.1	2.0	4.1	2.9	4.1	9.2	4.2	0.6	3.1	3.0	13.1	318.7
APR. 85	7.4	11.5	12.8	2.8	3.9	4.0	2.0	4.0	1.6	4.0	2.0	3.7	3.3	4.0	8.6	4.2	---	2.3	3.1	6.0	305.4
MAY 85	9.1	11.7	---	2.3	3.8	4.1	2.0	2.9	2.0	4.1	2.0	3.8	3.8	4.1	9.8	5.1	---	1.0	3.1	89.5	373.0
JUNE 85	10.1	10.4	14.0	2.9	3.9	4.0	1.6	1.9	1.9	3.4	2.8	4.0	4.0	3.7	8.9	4.9	---	1.0	4.0	92.2	380.0
JULY 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
AUG. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
SEP. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
OCT. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
NOV. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
DEC. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JAN. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
FEB. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
MAR. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
APR. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
MAY 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JUNE 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JULY 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
AUG. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
SEP. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
OCT. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
NOV. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
DEC. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JAN. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
FEB. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
MAR. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
APR. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
MAY 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JUNE 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JULY 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
AUG. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
SEP. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
OCT. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
NOV. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
DEC. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JAN. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
FEB. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
MAR. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
APR. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
MAY 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JUNE 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JULY 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
AUG. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
SEP. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
OCT. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
NOV. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0

Number of failures as a function of time & sqd. Sheet 1 of 2

Date	Sqr. Number 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
DEC. 84	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JAN. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
FEB. 85	---	---	---	---	1.0	---	---	---	---	---	---	---	---	---	---	---	---	---	---	1.0	---	---	---	---
MAR. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
APR. 85	---	2.0	---	---	---	---	2.0	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
MAY 85	---	---	---	---	---	---	---	---	---	---	---	---	---	2.0	---	---	---	---	---	---	---	---	---	---
JUNE 85	---	2.0	3.0	---	2.0	---	---	---	---	---	---	---	---	---	---	2.0	---	---	---	---	---	---	---	---
JULY 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
AUG. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
SEP. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
OCT. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
NOV. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
DEC. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JAN. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
FEB. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
MAR. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
APR. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
MAY 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JUNE 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JULY 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
AUG. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
SEP. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
OCT. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
NOV. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
DEC. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JAN. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
FEB. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
MAR. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
APR. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
MAY 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JUNE 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JULY 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
AUG. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
SEP. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
OCT. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
NOV. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
DEC. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JAN. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
FEB. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
MAR. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
APR. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
MAY 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JUNE 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JULY 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
AUG. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
SEP. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
OCT. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
NOV. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
Tot.	0.0	4.0	3.0	0.0	3.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	2.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
Adv.	0.0	0.6	0.4	0.0	0.4	0.0	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.0	0.3	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0

Number of failures as a function of time & sqd.

Date	Sqr. Number																				
	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	Adv.
DEC. 84	---	---	1.0	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	1.0
JAN. 85	---	---	---	---	---	---	---	---	---	---	---	3.0	---	---	---	---	---	---	---	---	3.0
FEB. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	2.0
MAR. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	2.0	---	2.0
APR. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	4.0
MAY 85	---	---	---	1.0	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	3.0
JUNE 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	9.0
JULY 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
AUG. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
SEP. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
OCT. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
NOV. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
DEC. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JAN. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
FEB. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
MAR. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
APR. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
MAY 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JUNE 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JULY 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
AUG. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
SEP. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
OCT. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
NOV. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
DEC. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JAN. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
FEB. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
MAR. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
APR. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
MAY 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JUNE 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JULY 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
AUG. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
SEP. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
OCT. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
NOV. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
DEC. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JAN. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
FEB. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
MAR. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
APR. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
MAY 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JUNE 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JULY 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
AUG. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
SEP. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
OCT. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
NOV. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
Tot.	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	
Adv.	0.0	0.0	0.2	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.0	

Number of failures per aircraft as a function of time & sqd. Sheet 1 of 2

Date	Sqr. Number 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
DEC. 84	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JAN. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
FEB. 85	---	---	---	---	0.1	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.3	---	---	---	---
MAR. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
APR. 85	---	0.1	---	---	---	---	0.3	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
MAY 85	---	---	---	---	---	---	---	---	---	---	---	---	---	0.2	---	---	---	---	---	---	---	---	---	---
JUNE 85	---	0.1	0.2	---	0.3	---	---	---	---	---	---	---	---	---	---	0.2	---	---	---	---	---	---	---	---
JULY 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
AUG. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
SEP. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
OCT. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
NOV. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
DEC. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JAN. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
FEB. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
MAR. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
APR. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
MAY 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JUNE 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JULY 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
AUG. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
SEP. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
OCT. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
NOV. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
DEC. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JAN. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
FEB. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
MAR. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
APR. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
MAY 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JUNE 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JULY 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
AUG. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
SEP. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
OCT. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
NOV. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
DEC. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JAN. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
FEB. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
MAR. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
APR. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
MAY 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JUNE 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
JULY 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
AUG. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
SEP. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
OCT. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
NOV. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
Tot.	0.0	0.2	0.2	0.0	0.4	0.0	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.2	0.0	0.0	0.0	0.3	0.0	0.0	0.0	0.0
Adv.	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Number of failures per aircraft as a function of time & sqd. Sheet 2 of 2

Date	Sqr. Number 25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	Adv.
DEC. 84	---	---	0.1	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JAN. 85	---	---	---	---	---	---	---	---	---	---	---	0.6	---	---	---	---	---	---	---	---	0.0
FEB. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
MAR. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.7	---	0.0
APR. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
MAY 85	---	---	---	0.4	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JUNE 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JULY 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
AUG. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
SEP. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
OCT. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
NOV. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
DEC. 85	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JAN. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
FEB. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
MAR. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
APR. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
MAY 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JUNE 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JULY 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
AUG. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
SEP. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
OCT. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
NOV. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
DEC. 86	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JAN. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
FEB. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
MAR. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
APR. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
MAY 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JUNE 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JULY 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
AUG. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
SEP. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
OCT. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
NOV. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
DEC. 87	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JAN. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
FEB. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
MAR. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
APR. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
MAY 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JUNE 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
JULY 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
AUG. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
SEP. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
OCT. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
NOV. 88	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	0.0
Tot.	0.0	0.0	0.1	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.7	0.0	
Adv.	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	

```
Max time in data        =   541   JUNE 85
Min time in data        =   338   DEC. 84
Total number of data    =    24
Number to be cal.       =    24
Mean time between fail.=   203.000
Mean value for unseen   =   203.000
Max time for cal. time  =   304.500
```

```
Time period from 1 to 12
log(tbf) versus log(t) fit gives
tbf = a*t**b
a   =   1.0086003765E+08
b   =  -2.7882376479E+00
====================
Time period from 13 to 24
log(tbf) versus log(t) fit gives
tbf = a*t**b
a   =   4.2291666667E+00
b   =   0.0000000000E+00
====================
Time period from 25 to 36
log(tbf) versus log(t) fit gives
tbf = a*t**b
a   =   4.2291666380E+00
b   =   9.9175391922E-10
====================
Time period from 37 to 48
log(tbf) versus log(t) fit gives
tbf = a*t**b
a   =   4.2291666078E+00
b   =   1.9426552115E-09
====================
Time period from 1 to 48
log(tbf) versus log(t) fit gives
tbf = a*t**b
a   =   2.7425572972E+05
b   =  -1.6349644806E+00
====================
```

* Monthly Failure Rates *

Data File used for study: 1R520A6.DTB
Date of study: 02/11/87

Date	Num. Fail	Adv. Inv.	Adv. Tbf	Ratio of Tbf/MTBF	Cal. Tbf	Delta cal-T	Var. Tbf	Delta var-T
DEC. 84	1	406	1718	8.46	1032	-685.8	7398	5680.1
JAN. 85	3	328	463	2.28	332	-130.8	1071	607.9
FEB. 85	2	362	766	3.77	171	-594.9	346	-420.3
MAR. 85	2	319	674	3.32	107	-566.9	155	-518.9
APR. 85	4	305	323	1.59	74	-248.6	83	-239.7
MAY 85	3	373	526	2.59	55	-470.7	50	-475.8
JUNE 85	9	380	179	0.88	43	-135.7	33	-146.0
JULY 85	0	0	4	0.02	34	30.2	22	18.2
AUG. 85	0	0	4	0.02	28	24.2	16	11.9
SEP. 85	0	0	4	0.02	24	19.7	12	7.8
OCT. 85	0	0	4	0.02	20	16.2	9	5.0
NOV. 85	0	0	4	0.02	18	13.5	7	3.0
DEC. 85	0	0	4	0.02	16	11.3	4	0.0
JAN. 86	0	0	4	0.02	14	9.6	4	0.0
FEB. 86	0	0	4	0.02	12	8.1	4	0.0
MAR. 86	0	0	4	0.02	11	6.9	4	0.0
APR. 86	0	0	4	0.02	10	5.8	4	0.0
MAY 86	0	0	4	0.02	9	4.9	4	0.0
JUNE 86	0	0	4	0.02	8	4.1	4	0.0
JULY 86	0	0	4	0.02	8	3.5	4	0.0
AUG. 86	0	0	4	0.02	7	2.9	4	0.0
SEP. 86	0	0	4	0.02	7	2.4	4	0.0
OCT. 86	0	0	4	0.02	6	1.9	4	0.0
NOV. 86	0	0	4	0.02	6	1.5	4	0.0
DEC. 86	0	0	4	0.02	5	1.1	4	0.0
JAN. 87	0	0	4	0.02	5	0.8	4	-0.0
FEB. 87	0	0	4	0.02	5	0.5	4	-0.0
MAR. 87	0	0	4	0.02	4	0.2	4	-0.0
APR. 87	0	0	4	0.02	4	-0.0	4	-0.0
MAY 87	0	0	4	0.02	4	-0.3	4	-0.0
JUNE 87	0	0	4	0.02	4	-0.5	4	0.0
JULY 87	0	0	4	0.02	4	-0.7	4	0.0
AUG. 87	0	0	4	0.02	3	-0.8	4	0.0
SEP. 87	0	0	4	0.02	3	-1.0	4	0.0
OCT. 87	0	0	4	0.02	3	-1.1	4	0.0
NOV. 87	0	0	4	0.02	3	-1.3	4	0.0
DEC. 87	0	0	4	0.02	3	-1.4	4	0.0
JAN. 88	0	0	4	0.02	3	-1.5	4	0.0
FEB. 88	0	0	4	0.02	3	-1.6	4	-0.0
MAR. 88	0	0	4	0.02	2	-1.7	4	-0.0
APR. 88	0	0	4	0.02	2	-1.8	4	-0.0
MAY 88	0	0	4	0.02	2	-1.9	4	-0.0
JUNE 88	0	0	4	0.02	2	-2.0	4	-0.0
JULY 88	0	0	4	0.02	2	-2.1	4	0.0
AUG. 88	0	0	4	0.02	2	-2.2	4	0.0
SEP. 88	0	0	4	0.02	2	-2.3	4	0.0
OCT. 88	0	0	4	0.02	2	-2.3	4	0.0
NOV. 88	0	0	4	0.02	2	-2.4	4	0.0
	24	52	100	0.49	44	176.7	195	842.8

APPENDIX G

MONTE CARLO ANALYSIS

[BLANK PAGE]

MONTE CARLO ANALYSIS

Spares Projection

The following algorithm was developed to project spare-parts replacement requirements and stockpile needs for aircraft components. This projection is based on a failure rate that can be a function of many variables. For the studies used to develop NAIL analytical functions, time was the only variable that had significant correlation; however, in general, this and other replacement equation parameters will be functions of several independent variables. The equation for replacement is given by

$$R_p = h \cdot [1 - F_r] \cdot N_a(t) \cdot [s(t) + r(v)],$$

where

R_p = project replacement rate (numbers per unit time),

F_r = fraction recycled,

$r(v)$ = failure rate (numbers per unit time),

$N_a(t)$ = number of aircraft,

h = number of components per aircraft,

$s(t)$ = stockpile factor,

t = time dependence,

v = multivariable dependence (i.e., $v = V_1, V_2, V_3$, etc.)

For the two components investigated as examples, the fraction recycled (F_r) could not be estimated because of the lack of depot-level data. Thus, this value was assumed to be zero. However, when the model is fully implemented, a value would be estimated for each component for which projections are required.

The failure rates would normally be a function of flight-hours, number of landings, etc. However, this study indicated that the tape transport and IFF demonstrated the best correlation with time. Therefore, standard reliability projection techniques were used. For ease of calculation, the failure rate was assumed to be the Weibull hazard function:

$$r(t) = \frac{\beta \, t^{\beta - 1}}{\sigma^{\beta}}$$

The number of aircraft is assumed to be a linear function of time, and, thus,

$$N_a(t) = N_o + \Delta N(t - t_o),$$

where

t_o = a base year from which numbers of aircraft are known,

N_o = the number of aircraft at base year,

ΔN = the number of aircraft added per unit time.

The term S(t) is a factor that will ensure that the number of parts will be sufficient to maintain some level of confidence that statistical fluctuations will not exhaust the stockpile between the replenishment of components in the inventory. The equation for this factor is

$$S(t) = \frac{gz}{y + 1}$$

where

z = one over the mode, or failure rate at the mode,

y = the number of times the stockpile is restocked, and

g = a factor that relates to the confidence that the parts supply will not be exhausted.

The value for g is a function of the required confidence and also may be a function of other variables. Figure G.1 shows the values resulting from a Monte Carlo study. Note that the solid line in the figure is the model projection and the small triangles represent samples of the random number.

Monte Carlo Simulation

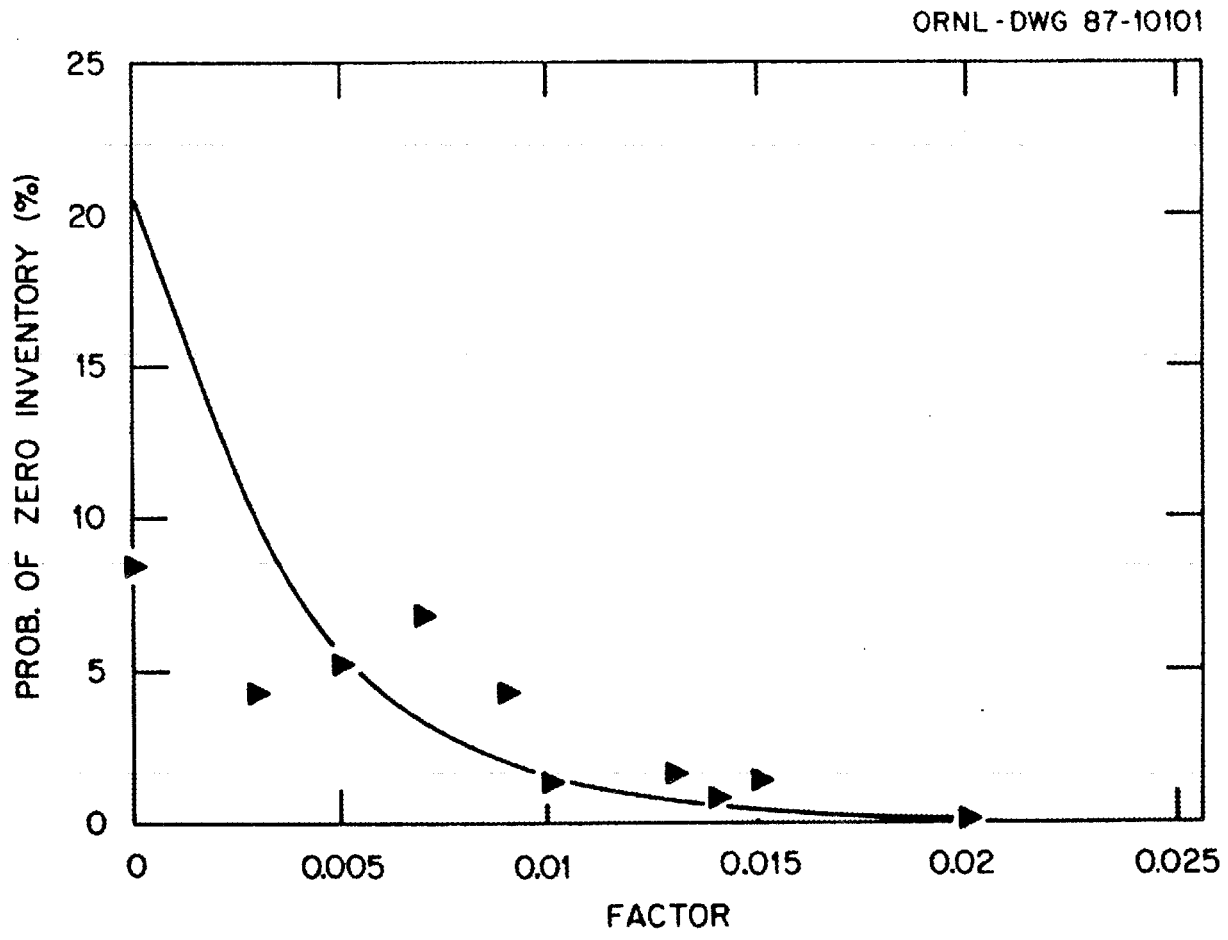
Monte Carlo simulations were used because the relationships of the variables in the model are too complex for analytic solutions. Monte Carlo simulation uses a computer to evaluate the model numerically over the time period of interest. Data are gathered to estimate the true characteristics of the model.

The Monte Carlo model was used to determine confidence intervals for the stockpile model and to project numbers of repairs at the organization (O), intermediate (I), and depot (D) levels. Figure G.2 shows the number of failures each month. The Weibull projection has been superimposed on this representation of failures.

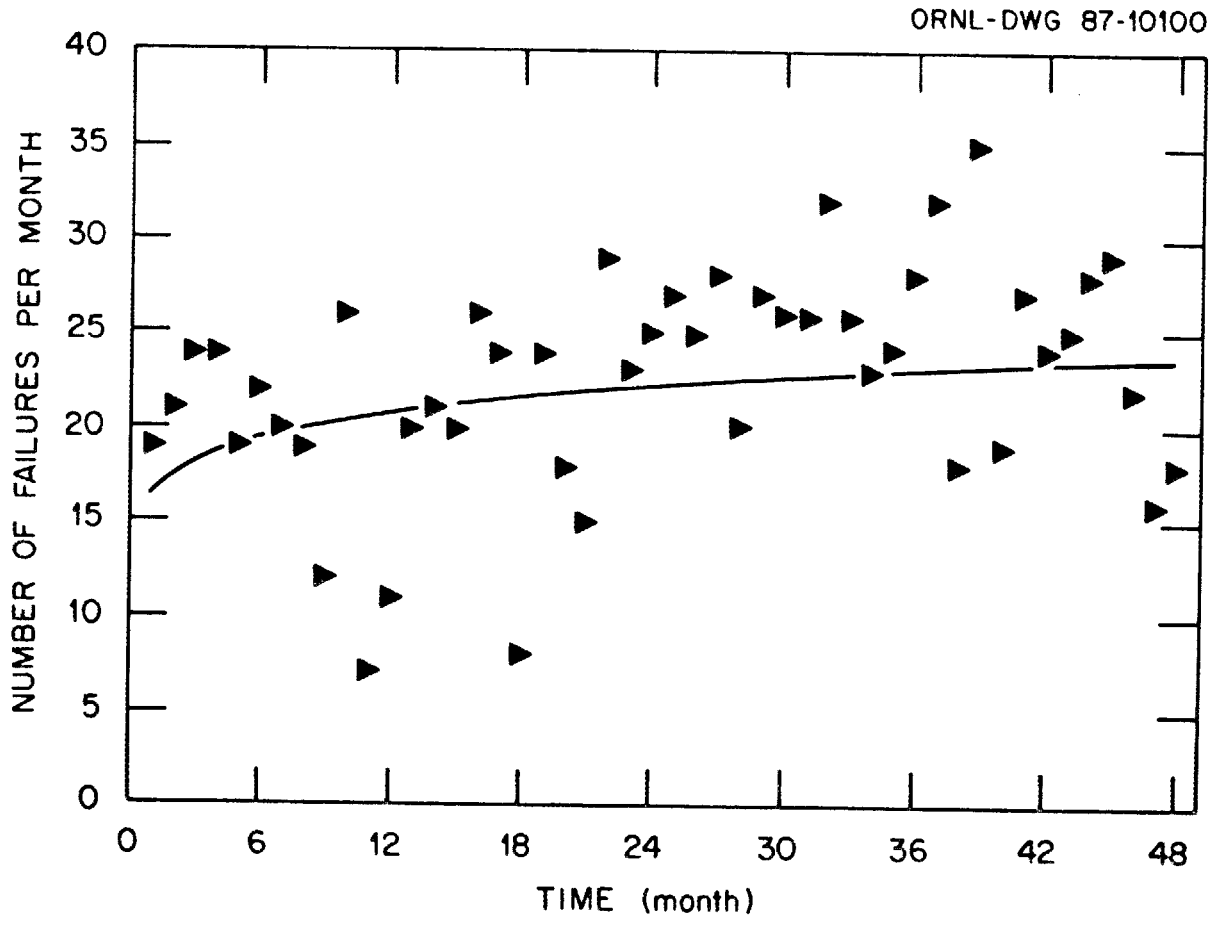
This projection may be used to estimate the number of required actions at each level by multiplying the factor g times the projection line.

Note that this contingency is constant with time and, therefore, contingency parts are acquired and spare parts do not continue to build up. The division of failure projections into the various maintenance levels was accomplished by studying the ratio of activities at each level over 4 years preceding the projection years. These ratios were approximately constant and were assumed to be invariant during the projection.

[BLANK PAGE]



Values Of Factor g For Adjustment Of Inventory
(IFF Receiver/Transmitter)



Number Of Failures Each Month Over 4-Year Period
(IFF Receiver/Transmitter)

APPENDIX H

NAIL: ON-LINE TUTORIAL

[BLANK PAGE]

NAIL: ON-LINE TUTORIAL

Introduction

The following is a print-out of the on-line tutorial provided with the NAIL package. This tutorial is displayed when the user types Run_Me.1st. The tutorial is also accessible at any time during NAIL operation via the Utility system (initiated by pressing the F2 key). The user can peruse the tutorial via this option and then return to the same location in his task without losing data or information.

[BLANK PAGE]

Naval Aviation Integrated Logistics

(NAIL)

Users Guide

This document describes version 1.0 of this program package.

R. C. DeLozier
Engineering Division
Oak Ridge National Laboratory

May 1986

Introduction

This collection of programs and data sets is version 1.0 of the Naval Aviation Integrated Logistics (NAIL) package. This package is a tool for logistics planning. Several analysis programs and data sets under the control of a simple menu program are used to manage and study inventories of spares. The NAIL users manual is meant to enable the user run the package. It does not describe the methods and techniques used in the analysis. This material is detailed in the Technical Users Guide.

This package has been checked for programming errors; however, as with any computer software, it is anticipated that this package will not perform as expected in some instances. This is more true for this release since this is the first version of NAIL. The user should contact the author (615-574-4009) if there are questions or problems concerning any step, calculated result or operation. A comment sheet for mailing to the program office can be created using the NAIL utility system.

User input forms the basis of almost all debugging and meaningful upgrades of programs. This input and questioning will also make the package more useful to the user and at the same time stimulate the development of improved versions of NAIL.

System Requirements

1) Hardware

The NAIL package requires a personal computer (e.g, the IBM PC or compatible) with at least 192K of memory. More core is desirable and if available should be used as a RAM or virtual disk. The package performs best with a hard disk and this is recommended; however, it will operate on a computer system with either a hard disk or with dual floppy disks. Any printer system should be compatible with NAIL; however, a dot matrix printer (such as an IBM or Epson) is recommended for quick draft level output. If plots are required as output, a Hewlett Packard (such as the HP 7475A) or compatible plotter is supported.

NOTE: Read/write speed is greatly enhanced by using RAM disks.

This alternate approach to disk input/output (IO) is available if the computer system has a large amount of accessible memory. The memory can be used as a large virtual disk such as VDISK. These virtual drives can be very large and transfer data very fast. On the other hand, the necessity for using this space for some program operations, such as multi-usable regression analysis, will likely limit many large IO data files to magnetic media.

2) Software Compatibility

This version of NAIL is intended for use with PC DOS or MS DOS. However, the package should install and execute correctly on almost all computer systems.

The package is compatible with most resident software; however, the ANSI device driver may interfere with screen colors. The current version should be compatible with all RAM disks as long as there is 192K free memory.

3) DOS Utilities

The controller diskette or the hard disk subdirectory must contain several DOS utilities or a path to these utilities. Before using the package on a dual floppy system, the following programs should be copied onto the NAIL package:

COMMAND.COM
MODE.COM (needed to set printer options)
EDLIN.COM (or any other editor)
BASICA.COM (needed for plotter interface).
VDISK.SYS (or any other RAM disk system)

Setting up your working diskettes

1) Backup Procedure

Before using the package you should copy the original diskettes onto a working set of diskettes or hard disk. The original diskettes will provide backup if the working set is damaged or corrupted. This backup procedure is accomplished by first formatting one blank diskette using the following command after the DOS prompt:

format b:/s

NOTE: the target diskette is in drive B.

Next, select the controller diskette. This diskette can be identified using the DIR command. The name "NAIL Contr." will appear in the volume field. Mark the new diskette as the Controller and copy the original onto it by placing the original in drive A:, the target diskette in drive B:, and entering:

copy a:*.* b:

The other diskettes are backed up similarly except the /s can be omitted from the format statement. This flag causes the operating system to be transferred to the diskette and only the controller requires this system.

2) Subdirectory Setup

If a hard disk is available, a subdirectory should be opened as follows:

```
md \NALC
cd \NALC
copy a:*.* \NALC
```

Using these commands, the source diskettes in drive A are read into the subdirectory and the third line (copy a:*.* \NALC) is repeated for each diskette copied to the hard disk. When actual data is available for analysis it can be stored using the same set of commands; however, it may be stored in a separate subdirectory such as P3DATA in order to simplify data management. This data subdirectory can be defined by:

md \P3DATA

Comments or data relevant to NAIL can be stored in files under this subdirectory.

3) Installing Utilities

To install the DOS utility files into NAIL use the COPY command. This command will be utilized after you receive the prompt A>. This prompt indicates that drive A is the currently active disk drive. Each command line for inputting the files will be followed by a return.

Use the IBM COPY command to install these files in the order listed:

```
copy command.com b:\nalc\
copy mode.com b:\nalc\
copy basica.com b:\nalc\
```

NOTE: b:\nalc\ is assumed to be the target drive and subdirectory

NOTE: If VDISK is used, consult the DOS reference manual for installation instructions.

The path command can be used efficiently on hard disk systems. For example, if the above DOS files reside in the root dictionary, the following command will permit NAIL to access these files:

```
PATH= C:\
```

Running the analysis package

The system is initiated by simply entering

NAIL

while in DOS and on the diskette or subdirectory containing the NAIL.EXE file. This call executes a controller program labeled DeMENU. This program can initiate hundreds of predefined procedures based on user options, sequences, and combinations of selections.

Alternately, the system can be initiated by entering the drive and path names as part of the command line. For example, if the data was stored on drive D in subdirectory \DATA the following call would indicate the location of the data:

NAIL D:\DATA

Even if the drive and subdirectory are input using this option, the NAIL system will first request the names of the disk drive and subdirectory which will be used for input and output data. The system will display what is currently assumed and the user can overwrite or accept the assumed drive and directory by pressing the Enter key. The user also inputs a general comment (e.g., F-14 readiness under wartime conditions) that will be used to label data produced during the session. This screen is exited by pressing the ESC key after the general comment is entered. The main interface screen is then displayed. This main user interface screen is shown in Figure 1.

In this figure, the user is presented with 51 procedures laid out in a 17 by 3 matrix. Each one of the steps will perform a separate operation, e.g. print a data file, regression analysis, etc. The user moves from one step to the next using the keyboard arrow keys. A window below the step matrix gives a brief text description of nine operations around and including the cursor step. This step is highlighted and displayed as the centered operation surrounded by eight adjacent steps. More information on the highlighted operation can be obtained by pressing the F1 help key.

NOTE: In NAIL, version 1.0, only 22 of the 51 available steps are defined. The remaining spaces are for future revisions and add-ons to the NAIL package.

The user initiates an operation by pressing the Enter key. After the selected operation is completed, control is returned to DeMenu. From this location, any of the steps can be executed as before.

Help Screens

NAIL provides several levels of assistance to the user. The F1 key will generally display detailed information to help the user regarding a pending operation or decision. For example, Figure 2 illustrates a sample step description message displayed when the F1 key is pressed.

Similarly, the user can access the utility screen from the DeMenu screen by pressing the F2 key. This screen is shown in Figure 3. From this utility screen the user can change the drive and directory used to input and output data; access the DOS system; and edit the steps. From the DeMenu screen, the user may edit the data comment files used to identify data sets, access the on-screen tutorial, and create commentary on the use of NAIL.

```
07:31:18                                  NAIL                                  05-07-1986
                                          DeMenu
1   4   7  10  13  16  19  22  25  28  31  34  37  40  43  46  49
2  [5]  8  11  14  17  20  23  26  29  32  35  38  41  44  47  50
3   6   9  12  15  18  21  24  27  30  33  36  39  42  45  48  51

                                   Steps
                 1. Remove report headings
                 2. Evaluate time between failures
                 3. Include R710 data
                 4. Regression analysis
                          ENTER TO EXECUTE
                            Plot data
                 6. Distribution analysis
                 7. Monte Carlo analysis
                 8. Monthly failure rate
                 9. Create plots of data

1.Use arrow keys to select step 3.For more details on a step use the F1 key
2.Execute step with  RETURN key 4.Utilities:use F2 key & to exit use Esc key
```

Figure 1: DeMENU User Interface Screen

```
07:31:18                                  NAIL                                  05-07-1986
                                          DeMenu
1   4   7  10  13  16  19  22  25  28  31  34  37  40  43  46  49
2   5   8  11  14  17  20  23  26  29  32  35  38  41  44  47  50
3   6   9  12  15  18  21  24  27  30  33  36  39  42  45  48  51

                          Details of Step 5
 This step generates plots of data created by other steps. The plot can
 display data in the form of bars, symbols, or lines using a HP plotter

 Hit any key to return to main menu

1.Use arrow keys to select step 3.For more details on a step use the F2 key
2.Execute step with  RETURN key 4.Utilities:use F2 key & to exit use Esc key
```

Figure 2: DeMenu User Interface Screen (Help Option)

```
Utilities
Change Drive/Dir
DOS Commands
NAIL Step Editor
Data Label Editor
NAIL Tutorial
Comment Sheet

Select item
F1 key: Description of item selected  Enter: Execute selection
```

Figure 3: DeMENU Utility Screen

Performing Analysis or Utility Operations

When a task has been selected using DeMENU, the NAIL package initiates one or more programs or computer functions required to accomplish it. These operations and programs will, in general, require input from the user via the keyboard and/or data files stored on magnetic media. The user will be queried and required to input information such as confidence levels, number of years, etc. The user must also select appropriate data file names for input and output at each analysis step.

Almost all of the programs requiring data file names as input will allow the user to view the files available in the selected area. This utility will attempt to position the most likely candidates at the front of the directory. The user will create new data files and should name these files with proper extensions so the package can properly position them in the directory. The naming conventions are simple. New files consisting of original R710 and R520 data should use DAT as a file name extension (R710P3.DAT, R520IFF.DAT, etc.). The first set of characters (maximum of eight) should be a name that describes the data set.

The various operations and programs will produce new data files using the input data files selected by the user. The program will name the new data files by retaining the first part of the name and incrementing the extension (e.g. R520IFF.DTA, R520IFF.DTB, etc.).

NOTE: There are two exceptions to this rule. All plot files (data to be plotted) and final reports will have an extension of Pxx, where xx is the number of the step that created it, e.g R710P3.P27, R20IFF.P02.

Program and Step Interaction

1) Data Preparation

Before using the NAIL package the user must store data in the form of the R520 and/or R710 report on machine-readable media. This will normally be done with a communication program such as PC-TALK. These data files should be stored under names using the extension, DAT, in order for the utility programs to recognize them. These data sets provide the primary input to the NAIL system as shown in Figures 4-6.

Figure 4 indicates the operations required to produce an MIR report. The operational steps for producing a Monthly Failure Rate Report are shown in Figure 5. The method for obtaining a Regression Analysis is shown in Figure 6.

2) Program Interaction

The reports shown in Figures 4-6 can be produced as a single operation by using the sequential operational steps 25-27, respectively. These steps call the operations indicated in the figures with minimum user input. Alternately, the user can perform each step by selecting the appropriate step using DeMenu. If this is done, it is important for the user to understand the interaction of programs because before a task, e.g., an MIR study, can be accomplished, other steps must be executed in order to create input needed to do the desired task. Figures 4-6 show the interaction of the operations and input and output at each step. The user interface is prompted from within a step and data files are enclosed with brackets, i.e. the symbols [...].

3) Program Steps

The program operations or steps used in the NAIL package are dynamic and are very easily modified and added to. The following are those steps included in version 1.0.

Step 1: Remove report headings

This step is normally the first step in the NAIL package. The input data file contains report headings, etc. that need to be removed before studies and analysis can be accomplished. The dates are changed to sequential numbers (1-1460 for four years) relative to the starting date of the program. This step requires input (R520 report) created outside of NAIL.

Step 2: Evaluate time between failures

This step simplifies the data for analysis. The aircraft tail numbers are used to find successive failures dates. When dates are identified, the difference is used to calculate the time between failures (TBF). This step requires input files created by step 1.

Step 3: Include R710 data

This step combines the step 2 file with the operational data file. The R710 data has been used to create a parameter data file (SUPERDR.SUP) that contains information on aircraft flight hours, landings, etc. This step requires input files created by steps 2, 8, and 13.

Step 4: Regression analysis

This step performs linear regression analysis on the file generated by step 3. The correlation equation is assumed to be linear and each parameter can be omitted to determine its effect. This program also can generate plots of data. This step requires input files created by step 19.

Step 5: Plot data

This step generates plots of data created by other steps. The plot can display data in the form of bars, symbols, or lines using an HP plotter, or compatible.

Step 6: Distribution analysis

This step evaluates and fits the failure data distribution function. The program also creates a temporary data file for use in Monte Carlo simulation of part inventory requirements. This step requires input files created by step 4.

Step 7: Monte Carlo analysis

This step does a simulation of the part inventory using the observed distribution function. This simulation indicates the stock required to reduce the probability of zero inventory to an acceptable confidence level. This step requires input files created by steps 6 and 8.

Step 8: Monthly failure rate

This step does an initial evaluation of the number of failures per month for a given time interval. The mean time between failures (MTBF) and other values are estimated. The monthly failure rate is fit to a Weibull projection equation. This step requires input files created by step 2.

Step 9: Create plots of data

This step is used to create plot files using data input from the keyboard and/or files specified by the user. Several equations can be fitted to the data and plotted as a line along with the input data.

Step 10: Find & count data

This program evaluates the number of occurrences of data in given columns of a data set. For example, this program is used to count the number of repairs at O, I, or D level.

Step 11: Combine two files

This program is used for combining two data sets. For example, the maximum time interval for the R520 report is two years; this program allows two such data sets to be combined into a four year set.

Step 12: Output data files

This program outputs data files to the printer along with column numbers. The output is used to find the column numbers of data fields for input to data management programs.

Step 13: Fix R710

This program is used to reduce the R710 report to a simplified data set. This step requires input files created outside of NAIL.

Step 14: Printer 132 columns

This step will change the output of Epson-type printers to 132 columns per line. This is required for most output in the NAIL system.

Step 15: Printer 80 columns

This step will change the printer output back to 80 column if needed.

Step 16: MIR Study

This analysis is used to project the number of removals and repairs at the O, I and D levels using a data set created by the step 2 activity.

Step 17: Slides for plotter

This step generates data files that can be plotted as text files. The HP plotter can be used to output slides with this utility.

Step 18: Set up RAM disk

This step assumes the use of SUPERDRV (AST) but the user can edit the QDELOGIC file and change SUPERDRV to any other procedure such as VDISK.

Step 19: Format data for regression analysis

This step changes the output file created by Step 2 to the compressed format required for input to step 4. This step can be used to remove time between failures having zero values.

Step 20: Sort file

This step is used to sort data files based on specified columns. If the R520 report is not sorted, this procedure can be used after step 1.

Step 25: MIR Report
The MIR Report is produced from the R710 and R520 data base during this step.

Step 26: Monthly Failure Rate

During this step, a report on the Monthly Failure Rate of a given set of parts is created from the R710 and R 520.

Step 27: Regression Analysis

A regression analysis is run utilizing the R710 and R520 reports at this program step.

Step 31: Monte Carlo Output Example

The Monte Carlo analysis is an repetitive and lengthy analysis that consists of many steps. This analysis projects the number of repairs at the O. I, and D level. The example shows the results of two such analyses.

FIGURE 4: MIR REPORT

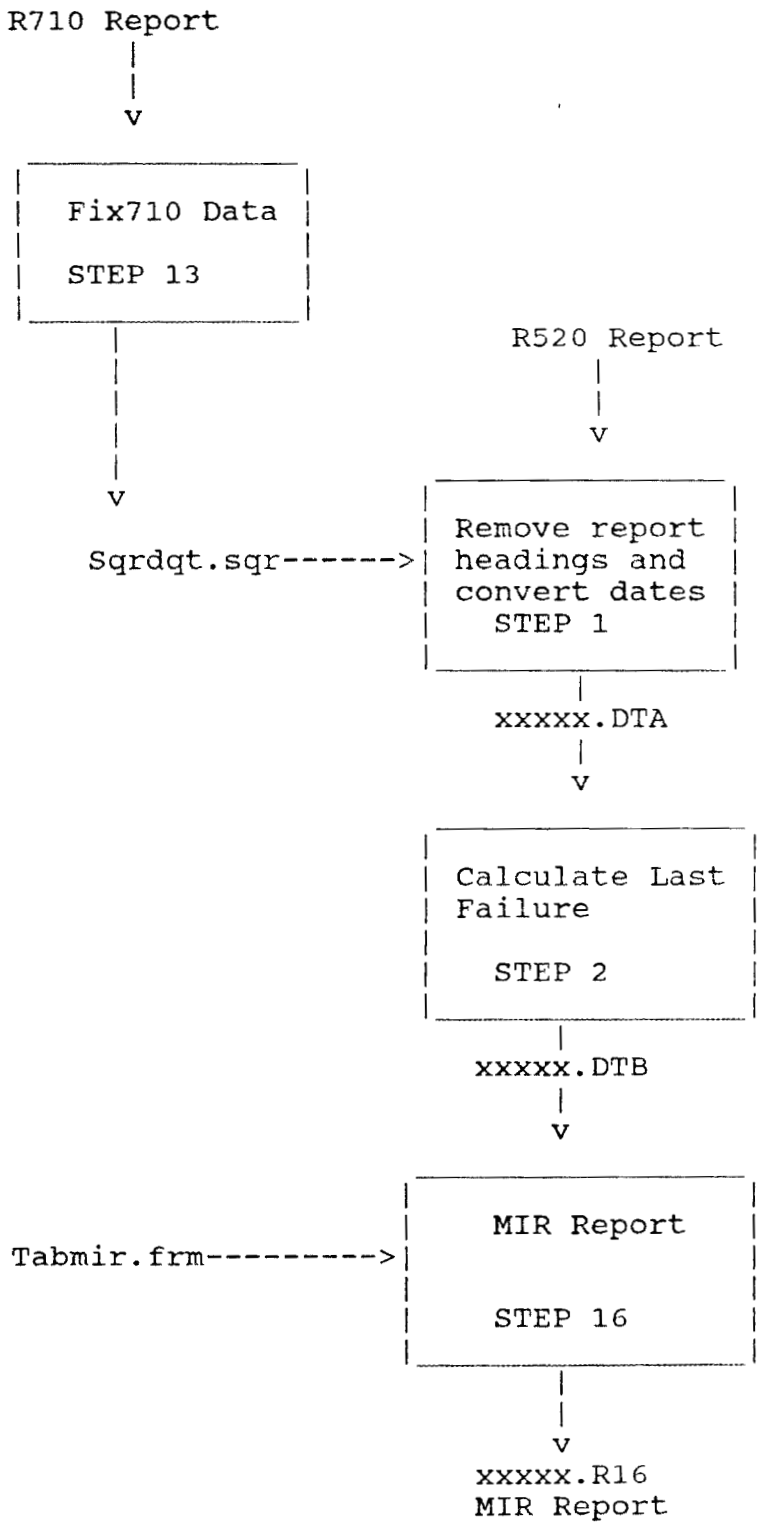


FIGURE 5: MONTHLY FAILURE RATE

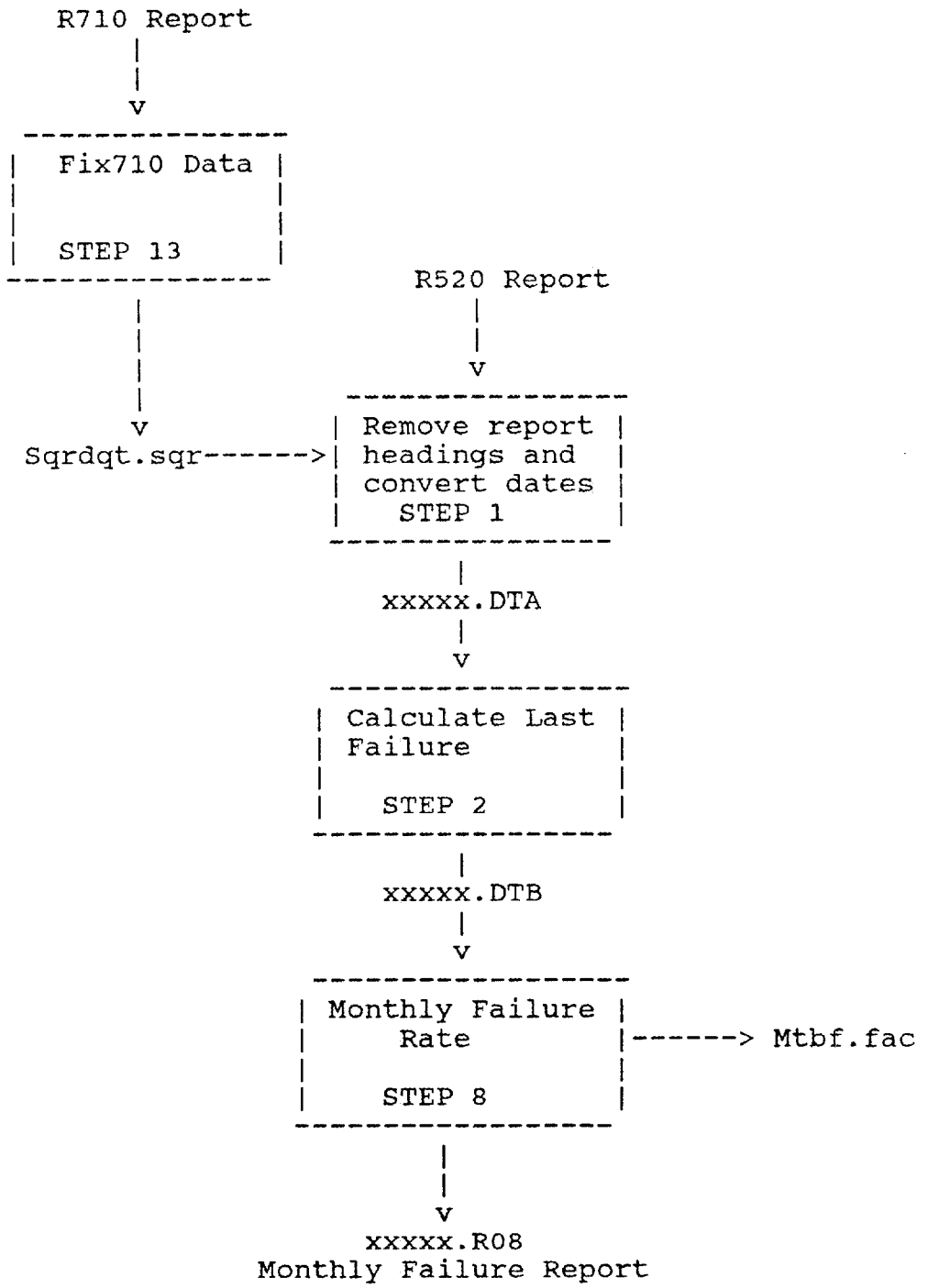
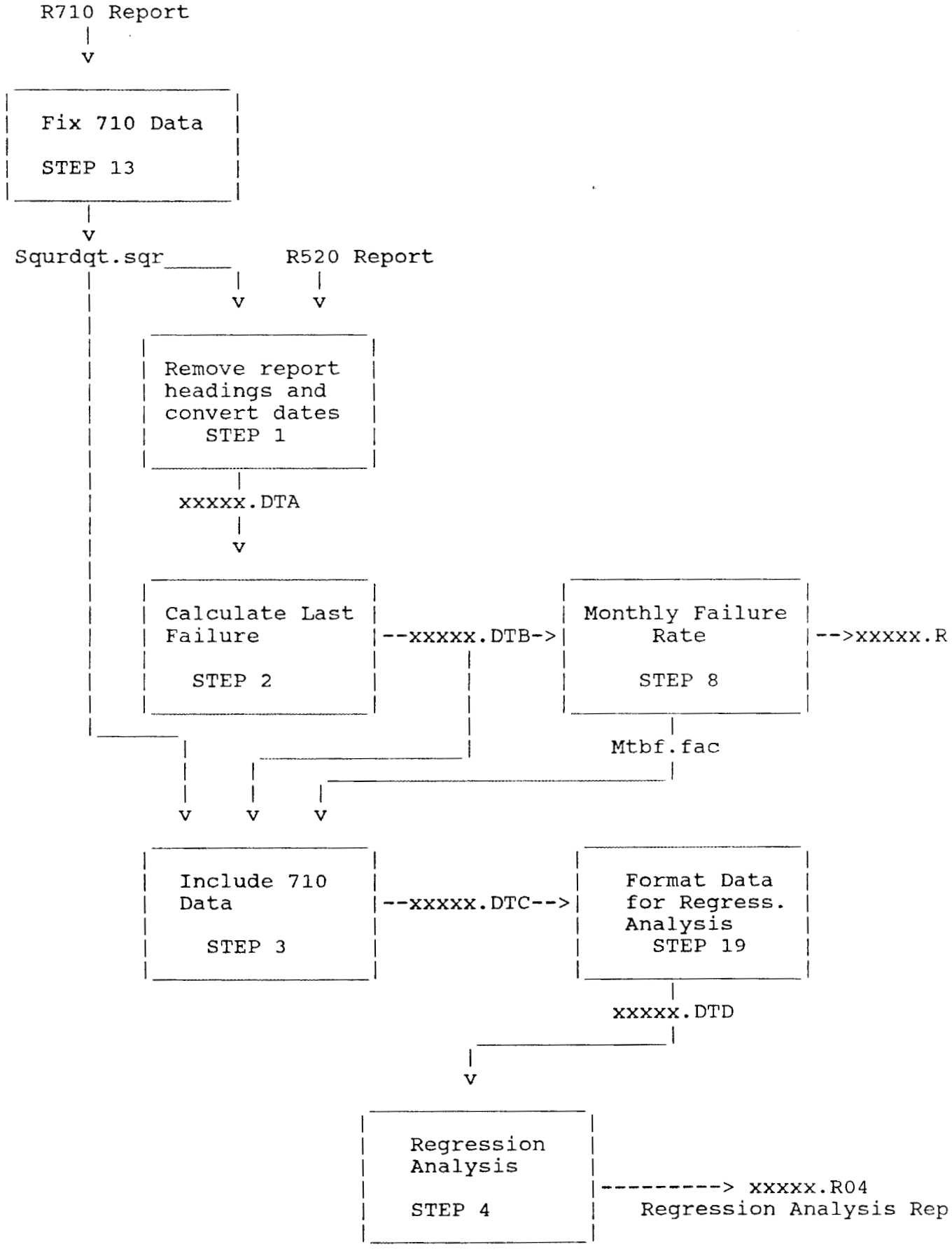


FIGURE 6: REGRESSION ANALYSIS



Example Data

The NAIL package contains several test data sets which will help the user learn how data files are input and output to various steps. There are several data files stored in two subdirectories (A6 and F18) on the sample data diskette. These files can be seen by typing cd A6, or cd F18, and entering DIR *.DAT. The following are the files in these directories:

DIRECTORY \A6

AIRCRAFT	WUC	FILENAME	BYTES
A6*	1113000	1R520A6.DAT	5725
	4284500	2R520A6.DAT	1595
	7691000	3R520A6.DAT	8114
	1413100	4R520A6.DAT	4791
A6**		R710A6.DAT	73984

DIRECTORY \F18

AIRCRAFT	WUC	FILENAME	BYTES
F18*	1431210	R520F18.DAT	36310
F18**		R710F18.DAT	56490

* Data for: Individual Maintenance Action Data Sets/For Date Range/Dec 84 Thru June 85

** Data for: Flight Activity And Inventory Data Sets/For Data Range/Dec 84 Thru June 85

In the F18 subdirectories there are two data sets. These data sets are supplied by the user for normal analysis. They are simply specific reports (R520 and R710) generated by the AMPAS system and stored as data on disk files. When these data sets are processed using the NAIL package the other data files are created by the various steps as shown in Figures 4-6. The user must run the steps in the sequence shown in these figures to create any data set shown as required input for another step, or use the series steps (e.g., 25, 26, and 27).

Using NAIL on a dual floppy system

As stated earlier, the NAIL package is designed to run most efficiently on a hard disk, or 1.2M floppy. However, NAIL will work on a system having two 360K floppy drives. The user will need to prepare a formatted diskette containing the DOS operating system (use the /s flag). After the new diskette is prepared, all programs required for a task must be installed on this diskette. The following programs must be installed for any task:

NAIL.EXE
QDELOGIC

The programs that are required to accomplish the particular analysis must be installed in addition to these files. The space restriction inherent in a 360K disk will limit the number of steps to around ten. The IO can be managed on the second drive using several disks. However, the largest data file will be limited to 360K.

APPENDIX I

SOURCE CODE SPECIFICATIONS: MAIN PROGRAMS & MODULES

[BLANK PAGE]

SOURCE CODES SPECIFICATIONS: MAIN PROGRAMS & MODULES

Introduction

This appendix lists the programs included in the NAIL package and their required support routines. Each program source code title is given and its operations described. The authors of each program are also identified. The other programs and subprograms that are required to compile the entire operation, the executable code name, and the necessary input files are listed. The final item is the name format of the operations output.

[BLANK PAGE]

Name: RmHead.pas
Title: Remove Report Headings (Step 1)
Description: Create data set from the R520 report. The headings are stripped off and dates are changed to sequential numbers.

Programmer: D.A.Holder
Source Code: RmHead.pas - - - - Main Program
Boxwin.pas - - - - Include Module
Help.pas - - - - - Include Module
AddDir.pas - - - - Include Module
Comments.pas - - - Include Module
Exists.pas - - - - Include Module
DiskSpace.pas - - Include Module
GetDir.pas - - - - Include Module
DirLst.pas - - - - Include Module
FileName.pas - - - Include Module
AddFileExt.pas - - Include Module
ReadField.pas - - Include Module
ReadCom.pas - - - Include Module
DisCom.pas - - - - Include Module
UpdateCom.pas - - Include Module
Executable Code: RmHead.com
Input Files: ??R520.Dat (* R520 Report *)
Sqrdqt.sqr (* Squadron Data Set *)
Output Files: xxxxxx.DTA

Name: LastFail.pas
Title: Evaluate Time Between Failures (Step 2)
Description: Calculate time between failure dates. When dates are identified, the difference is used to calculate the time between failures.
Programmer: D.A.Holder
Source Code: LastFail.pas - - - Main Program
Boxwin.pas - - - - Include Module
Help.pas - - - - - Include Module
AddDir.pas - - - - Include Module
Comments.pas - - - Include Module
Exists.pas - - - - Include Module
DiskSpace.pas - - Include Module
GetDir.pas - - - - Include Module
DirLst.pas - - - - Include Module
FileName.pas - - - Include Module
AddFileExt.pas - - Include Module
ReadField.pas - - Include Module
ReadCom.pas - - - Include Module
DisCom.pas - - - - Include Module
UpdateCom.pas - - Include Module
Executable Code: LastFail.com
Input Files: xxxxxx.DTA (* Output File produced by RmHead *)
Output Files: xxxxxx.DTB

Name: Include710.pas
Title: Include R710 Date (Step 3)
Description: Combine the data set created by LastFail with the operation data file.
Programmer: D.A.Holder
Source Code: Include710.pas - - Main Program
Boxwin.pas - - - - Include Module
Help.pas - - - - - Include Module
AddDir.pas - - - - Include Module
Comments.pas - - - Include Module
Exists.pas - - - - Include Module
DiskSpace.pas - - Include Module
GetDir.pas - - - - Include Module
DirLst.pas - - - - Include Module
FileName.pas - - - Include Module
AddFileExt.pas - - Include Module
Time.pas - - - - - Include Module
Date.pas - - - - - Include Module
ReadField.pas - - Include Module
ReadCom.pas - - - Include Module
DisCom.pas - - - - Include Module
UpdateCom.pas - - Include Module
Executable Code: Include7.com
Input Files: xxxxxx.DTB (* Output File produced by LastFail *)
Sqrdqt.sqr (* Squadron Data Set *)
Mtbf.fac
Output Files: xxxxxx.DTC

Name: FormatRa.pas
Title: Format Data for Regression Analysis (Step 19)
Description: Changes the Data set created by Include710 into a compressed format for input to the Regression Analysis (Step 4).
Programmer: D.A.Holder
Source Code: FormatRa.pas - - - Main Program
Boxwin.pas - - - - Include Module
Help.pas - - - - - Include Module
AddDir.pas - - - - Include Module
Comments.pas - - - Include Module
Exists.pas - - - - Include Module
DiskSpace.pas - - Include Module
GetDir.pas - - - - Include Module
DirLst.pas - - - - Include Module
FileName.pas - - - Include Module
AddFileExt.pas - - Include Module
ReadField.pas - - Include Module
ReadCom.pas - - - Include Module
DisCom.pas - - - - Include Module
UpdateCom.pas - - Include Module
Executable Code: FormatRa.com
Input Files: xxxxxx.DTC (* Output File produced by Include710 *)
Output Files: xxxxxx.DTD

Name: Regranl.pas
Title: Regression Analysis (Step 4)
Description: Performs linear regression analysis on the data set created by Include710 and produces the Regression Analysis report.
Programmer: D.A.Holder
Source Code: Regranl.pas - - - Main Program
Boxwin.pas - - - - Include Module
Help.pas - - - - - Include Module
AddDir.pas - - - - Include Module
Comments.pas - - - Include Module
Exists.pas - - - - Include Module
DiskSpace.pas - - Include Module
GetDir.pas - - - - Include Module
DirLst.pas - - - - Include Module
FileName.pas - - - Include Module
AddFileExt.pas - - Include Module
Date.pas - - - - - Include Module
Time.pas - - - - - Include Module
PrtComments.pas - Include Module
ReadField.pas - - Include Module
ReadCom.pas - - - Include Module
DisCom.pas - - - - Include Module
UpdateCom.pas - - Include Module
Executable Code: Regranl.com
Input Files: xxxxxx.DTD (* Output File produced by FormatRa *)
Output Files: xxxxxx.R04 (* Regression Analysis Report *)

Name: FixR710.pas
Title: Fix R710 (Step 13)
Description: Reduces the R710 report to a simplified data set.
Programmer: D.A.Holder
Source Code: FixR710.pas - - - Main Program
Boxwin.pas - - - - Include Module
Help.pas - - - - - Include Module
AddDir.pas - - - - Include Module
Comments.pas - - - Include Module
Exists.pas - - - - Include Module
DiskSpace.pas - - Include Module
GetDir.pas - - - - Include Module
DirLst.pas - - - - Include Module
FileName.pas - - - Include Module
AddFileExt.pas - - Include Module
ReadField.pas - - Include Module
ReadCom.pas - - - Include Module
DisCom.pas - - - - Include Module
UpdateCom.pas - - Include Module
Executable Code: Fixr710.com
Input Files: ??R710.Dat (* R710 Report *)
Output Files: Sqrdqt.sqr (* Squadron Data Set *)

Name: MonthFail.pas
Title: Monthly Failure Rate (Step 8)
Description: Evaluates the number of failures per month for a given time period and produces the Monthly Failure report. The mean time between failures is also estimated.
Programmer: D.A.Holder
Source Code: MonthFail.pas - - Main Program
Boxwin.pas - - - - Include Module
Help.pas - - - - - Include Module
AddDir.pas - - - - Include Module
Comments.pas - - - Include Module
Exists.pas - - - - Include Module
DiskSpace.pas - - Include Module
GetDir.pas - - - - Include Module
DirLst.pas - - - - Include Module
FileName.pas - - - Include Module
AddFileExt.pas - - Include Module
Date.pas - - - - - Include Module
Time.pas - - - - - Include Module
PrtComments.pas - Include Module
ReadField.pas - - Include Module
ReadCom.pas - - - Include Module
DisCom.pas - - - - Include Module
UpdateCom.pas - - Include Module
Executable Code: MonthFai.com
Input Files: xxxxxx.DTB (* Output File produced by LastFail *)
Output Files: xxxxxx.R08 (* Monthly Failure Report *)
Mtbf.fac (* Mean Time Between Failures *)

Name: Combine.pas
Title: Combine Two Filed (Step 12)
Description: Combines two data sets.
Programmer: D.A.Holder
Source Code: Combine.pas - - - Main Program
Boxwin.pas - - - - Include Module
Help.pas - - - - - Include Module
AddDir.pas - - - - Include Module
Comments.pas - - - Include Module
Exists.pas - - - - Include Module
DiskSpace.pas - - Include Module
GetDir.pas - - - - Include Module
DirLst.pas - - - - Include Module
FileName.pas - - - Include Module
ReadField.pas - - Include Module
ReadCom.pas - - - Include Module
DisCom.pas - - - - Include Module
UpdateCom.pas - - Include Module
Executable Code: Combine.com
Input Files: xxxxxx.D?? (* Any two Data Sets *)
Output Files: xxxxxx.D?? (* The Combined Data Sets *)

Name: MIR.pas
Title: MIR Study (Step 16)
Description: Projects the number of removals and repairs at the O,I, and D levels.
Programmer: D.A.Holder
Source Code: MIR.pas - - - - - Main Program
Boxwin.pas - - - - Include Module
Help.pas - - - - - Include Module
AddDir.pas - - - - Include Module
Comments.pas - - - Include Module
Exists.pas - - - - Include Module
DiskSpace.pas - - Include Module
GetDir.pas - - - - Include Module
DirLst.pas - - - - Include Module
FileName.pas - - - Include Module
AddFileExt.pas - - Include Module
Date.pas - - - - - Include Module
Time.pas - - - - - Include Module
PrtComments.pas - Include Module
ReadField.pas - - Include Module
ReadCom.pas - - - Include Module
DisCom.pas - - - - Include Module
UpdateCom.pas - - Include Module
Executable Code: MIR.com
Input Files: xxxxxx.DTB (* Output File produced by LastFail *)
Output Files: xxxxxx.R16 (* MIR Report *)
Mtbf.fac (* Mean Time Between Failures *)

Name: StpSort2.pas
Title: Sort File (Step 20)
Description: Sort data sets based on specified columns.
Programmer: A.E.Shanks
Source Code: StpSort2.pas - - - Main Program
Boxwin.pas - - - - Include Module
Help.pas - - - - - Include Module
AddDir.pas - - - - Include Module
Comments.pas - - - Include Module
Exists.pas - - - - Include Module
DiskSpace.pas - - Include Module
GetDir.pas - - - - Include Module
DirLst.pas - - - - Include Module
FileName.pas - - - Include Module
AddFileExt.pas - - Include Module
Executable Code: StpSort2.com
Input Files: xxxxxx.D?? (* Any Data Set *)
Output Files: xxxxxx.D?? (* Sorted Data Set *)

Name: FindCount.pas
Title: Find and Count Date (Step 10)
Description: Evaluates the number of occurrences of data in given columns of a data set. The results are sent directly to the printer.
Programmer: D.A.Holder
Source Code: FindCount.pas - - Main Program
Boxwin.pas - - - - Include Module
Help.pas - - - - - Include Module
AddDir.pas - - - - Include Module
Comments.pas - - - Include Module
Exists.pas - - - - Include Module
DiskSpace.pas - - Include Module
GetDir.pas - - - - Include Module
DirLst.pas - - - - Include Module
FileName.pas - - - Include Module
Date.pas - - - - - Include Module
Time.pas - - - - - Include Module
Executable Code: FindCount.com
Input Files: xxxxxx.D?? (* Any Data Set *)

Name: DocList.pas
Title: Document List
Description: Provides on-line or hard-copy user documentation.
Programmer: R.C.DeLozier
Source Code: DocList.pas
Executable Code: DocList.com
Input Files:
Output Files:

Name: Deedit.pas
Title: Edit Qdelogic.
Description: Allows user to edit the QDELOGIC data file.
Programmer: R.C.DeLozier
Source Code: Deedit.pas
Executable Code: Deedit.com
Input Files: QDELOGIC
Output Files:

Name: Data_Dir.pas
Title: Define Default Data Drive/Directory.
Description: Defines the Drive and/or Directory in which the data resides.
Programmer: D.A.Holder
Source Code: Data_Dir.pas - - - - Main Program
Boxwin.pas - - - - - Include Module
Help.pas - - - - - - Include Module
Exists.pas - - - - - Include Module
ValDir.pas - - - - - Include Module
Break.pas - - - - - Include Module
ReadField.pas - - - Include Module
Executable Code: Data_Dir.com
Input Files:
Output Files:

Name: ComSh.pas
Title: Comment Sheet.
Description: Allows user to enter their comments concerning the NAIL System and produces a hard copy of the comments entered.
Programmer: D.A.Holder
Source Code: ComSh.pas - - - - - Main Program
Boxwin.pas - - - - - Include Module
Date.pas - - - - - - Include Module
Help.pas - - - - - - Include Module
ReadField.pas - - - Include Module
Executable Code: ComSh.com
Input Files:
Output Files:

[BLANK PAGE]

APPENDIX J

SOURCE CODE LISTINGS

PROLOG Source Code

Appendix J.1	NAIL Controller	J-4

PASCAL Source Code

Appendix J.2	AddDir	J-15
Appendix J.3	AddFileExt	J-16
Appendix J.4	BoxWin	J-17
Appendix J.5	DisplayDir	J-18
Appendix J.6	Combine	J-20
Appendix J.7	Comments	J-24
Appendix J.8	ComSh	J-26
Appendix J.9	Data_Dir	J-32
Appendix J.10	Date	J-40
Appendix J.11	DeEditor	J-41
Appendix J.12	Dirlst	J-55
Appendix J.13	Diskspace	J-63
Appendix J.14	run_me.1st	J-64
Appendix J.15	Exists	J-67
Appendix J.16	FileName	J-68
Appendix J.17	FindCount	J-72
Appendix J.18	FixR710	J-82
Appendix J.19	FormatRA	J-87
Appendix J.20	GetDirectory	J-93
Appendix J.21	Help	J-97
Appendix J.22	Include710	J-98

[BLANK PAGE]

Appendix J.23	LastFail	J-108
Appendix J.24	MIR	J-116
Appendix J.25	MonthFail	J-133
Appendix J.26	PrtComments	J-156
Appendix J.27	ReadCom	J-158
Appendix J.28	ReadField	J-160
Appendix J.29	RegraAnl	J-163
Appendix J.30	RmHead	J-176
Appendix J.31	SortStep	J-184
Appendix J.32	Time	J-191
Appendix J.33	UpDateCom	J-192
Appendix J.34	ValDir	J-195

BASICA Source Code

Appendix J.35	HPDRAW	J-196
Appendix J.36	HPPLOT	J-201
Appendix J.37	MONTE	J-210
Appendix J.38	STATDIS5	J-212

[BLANK PAGE]

SOURCE CODE LISTINGS

Introduction

This appendix contains the source code listings of the programs and subroutines used in the Version 1.0 NAIL package. These programs are listed in alphabetical order under one of the three categories of programming languages in which they are written: PROLOG; PASCAL; BASICA. The user should reference Appendix I to determine the relation of these programs and subroutines to each NAIL function and/or the inter-relationship of the routines. Comments are embedded in each program as an aid toward understanding various operations.

[BLANK PAGE]

Appendix J.1

NAIL Controller

```
/*                                                                    */
/*                                                                    */
/*          NAIL: Naval Aviation Integrated Logistics                 */
/*                                                                    */
/*                    PROLOG Source Code                              */
/*                          by                                        */
/*                     R. C. DeLozier                                 */
/*                                                                    */
/*____________________________________________________________________*/

domains
  i       = integer
  s,a     = string
  c       = char
  x       = real
  sb      = symbol
  n       = i*
  l       = S*
  file    = dap;qq;qqq


PREDICATES
  Loo(i,i)
  NOTE(I)
  helpmenu                /* Starts help system                          */
  BLINK(i,i,i,i,i)        /* Causes menu item to blink at cursor position */
  proces(i)               /* What to do when help item selected          */
  menu(i,i,STRING,l,i)    /* Displays help menu                          */
  menu1(i,l,i,i,i)
  menu2(i,l,i,i,i,i)
  readkey(i)              /* Returns significant ASCII code after read   */
  infix(i,i,i)            /* Returns significant ASCII code of two       */
  turn(i,i,i)             /* Returns value of menu item after arrow key pressed */
  READC(CHAR,I,I)         /* Reads chracters, first and second ASCI code */
  READC2(I,I)             /* Reads the second ASCII code of control      */

  DO(I,I,l)               /* Executes menu item I                        */
  call_help(i,l,i,i)      /* Exits NAIL to DOS, Editor, Tutorial         */
  maxlen(l,i,i)           /* Finds the length of the longest string      */
  listlen(l,i)            /* Finds the length of a list                  */
  ritelist(i,i,l)         /* Used by the menu predicate                  */
```

```
  repeat                    /* Makes backtrack points                          */
  move(i,i,l,l,l,i,i)       /* Moves cursor to new item                        */
  move2(i,i,l,l,l,i,i)
  addx(n,n,i)               /* Names list of numbers (1,2, .. ,51)             */
  readlist(L,L,L)           /* Reads the main menu file (QDELOGIC)             */
  readit(l,l,l,l,l,l,i)     /* Reads one line of file into list until eof      */

  writexy(i,i)              /* Writes menu number at fixed loc. x,y            */
  loc(i,i,i)                /* Locates an item's x,y                           */
  writelist(n,i,i)          /* Writes the item numbers for the first time      */
  writedate(i)              /* Writes date in date window                      */
  writetime(i)              /* Writes time in time window                      */
  WLIST(L,I,I,i)            /* Writes the step window                          */
  WRITD(L,S,I,I,i,S)        /* Writes NINE steps around selected item          */
  lim(i,i)                  /* Sets number of items in list for step output    */
  BUMP(I,I)
  NUL(I,I)
  SPLIT(I,S,S,S)            /* Splits string into 2 strings less than I        */
  wrneat(s,i)               /* Writes a string that ends on blanks             */
  wxlist(L,i,i)             /* Writes the details                              */
  wrteC(s,i,i,i)            /* Writes a centered string at row                 */
  run(L,L,L,I)              /* Runs the system                                 */
  start(L,L,L,I)            /* Startup                                         */

  help
  BREAK(A,A,A,I)            /* Breaks a string into 2 strings split by ASCII code */
  break_up(A,I,I,I)
  SEP3(A,A,A,A)             /* Breaks up a string into three strings           */
  listit(L)                 /* Lists a list                                    */

goal
    comline(M),
    PRG="Data_Dir ** ",
    concat(PRG,M,PP),
    write(" NAIL Set Up "),
    write(PP),
    system(PP),
    start(L1,L2,L3,N),
    run(L1,L2,L3,N).
CLAUSES

    NUL(A,B) IF A<46, B=9.
    NUL(A,B) IF B=51-A.
```

```
    listit([]) if !.
    listit([T|L]) if
    write(T),nl,
    listit(L).

    SEP3(T,T1,T2,T3) IF
    BREAK(T,T1,TX,94),
    BREAK(TX,T2,T3,124).

    BREAK(T,T1,T2,WHAT) IF
    STR_LEN(T,L),
    BREAK_UP(T,L,WHAT,WHERE),W=L-WHERE,
    FRONTSTR(W,T,T1,TX2),
    FRONTCHAR(TX2,_,T2).

    break_up(_,0,_,0).
    break_up(T,L,WHAT,WHERE) if
    frontCHAR(T,F,TN), CHAR_INT(W,WHAT),
    F<>W,
    LX=(L-1),
    BREAK_UP(TN,LX,WHAT,WHERE),!.
    BREAK_UP(_,L,_,L).

maxlen([H|T],MAX,MAX1) if
        str_len(H,LEN),
    LEN>MAX,!,
    maxlen(T,LEN,MAX1).
maxlen([_|T],MAX,MAX1) if
        maxlen(T,MAX,MAX1).
maxlen([],LEN,LEN).

readlist(L1,L2,L3) if openread(qq,"qdelogic"),
readdevice(KEY),
readdevice(qq),
readit([],[],[],L1,L2,L3,51),     closefile(qq),
readdevice(KEY).

readit(L1,L2,L3,L1,L2,L3,_) if eof(qq).
readit(L1,L2,L3,L1,L2,L3,0) .

readit(L1,L2,L3,A,B,C,I) if
readln(L),  In=I-1,
SEP3(L,TA1,TA2,TA3),
T1=TA1,T2=TA2,T3=TA3,
readit([T1|L1],[T2|L2],[T3|L3],A,B,C,In).

wlist([],_,_,_) if !.
```

```
wlist([H|L],Is,IsT,Mx) if
Is<=Ist+5,
shiftwindow(6),clearwindow,
ISN=IS-1,
lim(V,Ist),
writd(L,Ch,V,Isn,Ist,H),
str_len(CH,Ih),
SHIFTWINDOW(5),clearwindow,
attribute(15),
D=Mx-Ih,K=D/2+1,
cursor(0,K),
write(CH).

wlist([_|L],This,Is,Mx) if
Thisn=This-1,
wlist(L,Thisn,Is,Mx).

lim(V,I) if I<47,V=9.
lim(V,I) if V=55-I.

writd([H|L],CH,C,IS,IST,F) IF
C>0,
IS=IST,
CH=H,
CN=C-1,ISN=IS-1,
WRITD(L,CH,CN,ISN,IST,F).
WRITD([H|L],Ch,C,IS,IST,F) IF C>0,
CN=C-1,ISN=IS-1,BUMP(C,CB),cursor(CB,0),
WRITE(IS,". ",H),
WRITD(L,CH,CN,ISN,IST,F).

writd(_,A,0,_,_,_) if bound(A).
writd([],A,_,_,_,_) if bound(A).
writd(_,A,_,_,_,F) if A=F.

BUMP(A,B) IF A>4,B=A+1.
BUMP(A,B) IF B=A-1.

wrtec(S,L,W,R) IF
D=W-L,
C=D/2,
CURSOR(R,C),
WRITE(S),NL.

WXLIST([],_,_) IF !.
WXLIST([H|_],IS,IS) IF
STR_LEN(H,L),HH=H,
```

```
WRNEAT(HH,L),!.
WXLIST([_|L],THIS,THAT) IF
T=THIS-1,
WXLIST(L,T,THAT).

WRNEAT(H,L) IF L<73,
WRiTE(H),NL.

WRneat(H,_) IF
SPLIT(71,H,F,B),
WRITE(F),NL,
STR_LEN(B,LB),
WRNEAT(B,LB).

SPLIT(0,S,S,"").
SPLIT(I,S,F,B) IF
FRONTSTR(I,S,F,B),
FRONTCHaR(B,C,BB),
CHAR_INT(C,CX),
CX=32,
split(72,BB,_,_).
SPLIT(I,S,F,B) IF
IN=I-1,
SPLIT(IN,S,F,B).

BLINK (B,C2,B,_,C2) .
BLINK(_,C1,_,C1,_) .

loc(AH,X,YY) if
Y=AH-1 AND
YY=Y MOD 3 AND
X=2+(AH -YY)*1.333   .

addx(Na,Na,1).

addx(N1,N2,X) if
XN=X-1
and addx([XN|N1],N2,XN).

writelist([],_,_)
.
writelist([H|T],I,AT) IF
IH=H AND  loc(IH,X,Y)
and cursor(Y,X)
AND IN=I+1
AND blink(IH,K,AT,31,159)
and writexy(K,IH)
and writelist(T,IN,AT).
```

```
writedate(I) if
shiftwindow(109)
and date(Yr,Mon,Day)
and writef("%2%1%2%1%4",Mon,"/",Day,"/",Yr)
and shiftwindow(I).

writetime(I) if
shiftwindow(108)
and clearwindow
and time(Hr,Min,Sec,_)
and writef("%3%1%2%1%2",Hr,":",Min,":",Sec)
and shiftwindow(I).

writexy(A,V) if
loc(V,X,Y) and
cursor(Y,X) and
attribute(A) and
writeF("%2",V) .

start(Menu_short,Menu_Details,Menu_Do,Num_List) if
 clearwindow,
 readlist(Menu_short,Menu_Details,Menu_Do)  ,
 listlen(Menu_short,Num_List).

run(Ms,Mx,Md,Num_List) if
  AT=5
  AND MAXLEN(Ms,0,MsX),MsX2=MsX+4,CC1=80-Msx2,CC=CC1/2,MB=MSX2+1,CCC=CC+2
  AND MAKEWINDOW(6,7,0,"",9,CCc,11,MB)

  and makewindow(105,96,0,"",23,0,2,80)
  and nl,write(" 1.Use arrow keys to select step   3.For more details on a step use the  F1 key")
  and nl,write(" 2.Execute step with Enter\17\217 key   4.Utilities use F2 key & to exit use Esc key")
  and makewindow(102,14,14,"",0,35,3,10)
  and write("  NAIL")
  and makewindow(106,30,14,"\217 DeMenu \192",2,0,21,80)
  and makewindow(109,14,0,"",1,68,1,11)
  and makewindow(108,14,0,"",1,1,1,20)
  and writetime(106)
  and writedate(106)
  AND MAKEWINDOW(107,31,0,"",4,3,3,74)
  and cursor(1,1)
  and addx([],Numbers,52)
  and writelist(Numbers,1,At)
  and makewindow(106,15,15,"Steps",8,3,13,74)
  AND MAKEWINDOW(5,15,14," Enter \17\196\217 to execute ",13,CC,3,Msx2)
  AND SHIFTWINDOW(107)
  and move(AT,_,Ms,Mx,Md,Num_List,MsX)
```

```
  AND SHIFTWINDOW(105)
  , removewindow, makewindow(105,30,0,"",0,0,25,80).

listlen([],0).
listlen([_|T],N):-
        listlen(T,X),
        N=X+1.

ritelist(_,_,[])
.
ritelist(LI,ANTKOL,[H|T]):-field_str(LI,0,ANTKOL,H),
LI1=LI+1,ritelist(LI1,ANTKOL,T).

repeat. repeat:-repeat.

readkey(N0) if
readc(_,N1,N2),
infix(N0,N1,N2).

infix(N2,0,N2).
infix(N1,N1,_).

turn(AT,72,ATN) if AT > 1 and ATN = AT - 1.
turn(AT,72,ATN) if ATN = AT + 50.
turn(AT,80,ATN) if AT < 51 and ATN = AT + 1.
turn(AT,80,ATN) if ATN = AT - 50.
turn(AT,75,ATN) if AT > 3  and ATN = AT - 3.
turn(AT,75,ATN) if ATN = AT + 48.
turn(AT,77,ATN) if AT < 49 and ATN = AT + 3.
turn(AT,77,ATN) if ATN = AT-48.
turn(AT,73,ATN) if AT > 15 and ATN = AT -15.
turn(AT,73,ATN) if ATN = 36+AT.
turn(AT,81,ATN) if AT < 37 and ATN = AT +15.
turn(AT,81,ATN) if ATN = AT-36.
turn(_,71,ATN) if ATN = 1.
turn(_,79,ATN) if ATN = 51.
TURN(AT,_,AT).

READC(C,N1,N2) IF READCHAR(C) ,
CHAR_INT(C,N1),
READC2(N1,N2).
READC2(13,0) IF !.
READC2(0,N2) IF READCHAR(C2) ,
CHAR_INT(C2,N2).
READC2(_,0).
```

```
move(AT,ATN,M,Mx,Md,ML,MsX) if
WRITETIME(106),attribute(7),
wlist(M,ML,AT,MsX),
SHIFTWINDOW(107),
loc(AT,X,Y),
cursor(Y,X),
READC(_,N1,N2), call_help(At,Mx,ML,N2),
turn(AT,N2,ATN),
writexy(159,ATN),
WRITEXY(31,AT),MOVE2(ATN,N1,M,Mx,Md,ML,MsX).

call_help(_,_,_,60) if help.
call_help(At,Mx,ML,59) if
   str_int(NN,At),concat(NN," ",NNN),
   concat(" Details of step ",NNN,Tx),
   makewindow(36,15,14,Tx,8,3,13,74),
   CURSOR(10,1),
   attribute(112),
   WRITE("          Hit any key to return to the main menu          "),
   CURSOR(1,0),attribute(15),
   wXlist(Mx,ML,At),readchar(_),
   removewindow.
call_help(_,_,_,_).

MOVE2(_,27,_,_,_,_,_) IF !.
Move2(AT,13,M,Mx,Md,ML,MsX) if DO(AT,ML,Md) , writexy(159,AT) ,
MOVE(AT,_,M,Mx,Md,ML,MsX).

Move2(AT,_,M,Mx,Md,ML,MsX) if AT<52 AND AT>0 AND
move(AT,_,M,Mx,Md,ML,MsX).

MOVE2(_,_,M,Mx,Md,ML,MsX) IF MOVE(5,_,M,Mx,Md,ML,MsX).

DO(_,_,[]).
DO(IS,IS,[H|_]) IF
SYSTEM(H).
DO(IST,IS,[_|L]) IF ISN=IS-1,
DO(IST,ISN,L).

menu(LI,KOL,TXT,LIST,CHOICE) if
shiftwindow(121),
maxlen(LIST,0,ANTKOL),
listlen(LIST,LEN),ANTLI=LEN,LEN>0,
HH1=ANTLI+2,HH2=ANTKOL+2,
makewindow(123,31,15,TXT,LI,KOL,HH1,HH2),
HH3=ANTKOL,
ritelist(0,HH3,LIST),cursor(0,0),
```

```
menu1(0,LIST,ANTLI,ANTKOL,CH),
CHOICE=1+CH.

menu1(LI,LIST,MAXLI,ANTKOL,CHOICE) if
   field_attr(LI,0,ANTKOL,112),
   cursor(LI,0),
   readkey(KEY),loo(LI,KEY),
   menu2(LI,LIST,MAXLI,ANTKOL,CHOICE,KEY).

Loo(CH,F59) IF F59>58, F59<61,
MAKEWINDOW(111,112,14," Press space bar to return to utilities ",2,2,5,76),
NOTE(CH),
READKEY(_),
REMOVEWINDOW,!.
loo(_,_).

menu2(_,_,_,_,-1,27) if !.
menu2(LI,_,_,_,CH,13) if !,CH=LI.

menu2(LI,LIST,MAXLI,ANTKOL,CHOICE,72) if
   LI>0,!,
   field_attr(LI,0,ANTKOL,31),
   LI1=LI-1,
   menu1(LI1,LIST,MAXLI,ANTKOL,CHOICE).

menu2(LI,LIST,MAXLI,ANTKOL,CHOICE,80) if
   LI<MAXLI-1,!,
   field_attr(LI,0,ANTKOL,31),
   LI1=LI+1,
   menu1(LI1,LIST,MAXLI,ANTKOL,CHOICE).

menu2(LI,LIST,MAXLI,ANTKOL,CHOICE,_) if
   menu1(LI,LIST,MAXLI,ANTKOL,CHOICE).


HELP if
      makewindow(11,31,15,"",0,0,25,80),
   makewindow(121,112,0,"",22,1,2,78),
   write(" Esc Key: Return to NAIL                Arrow keys (\24 \25): Select item      "),
   write(" F1  Key: Description of item selected    Enter  (\17\196\217): Execute selection"),
   helpmenu.

helpmenu if   repeat,
         menu(6,30," Utilities ",
```

```
        [ "",
          " Change Drive/Dir ",
          " Dos Commands ",
          " NAIL Step Editor ",
          " Data Label Editor ",
          " NAIL Tutorial ",
          " Comment Sheet ",
          ""],CHOICE},
      proces(CHOICE),
      CHOICE=0,!,removewindow,
      removewindow,removewindow.


proces(0).
proces(1).
proces(2) if system("Data_Dir"),!.
proces(3) if system(""),!.
proces(4) if SYSTEM("DeEDIT"),!.
proces(5) if
      CAL="DeEdit DataList.cmt",
   SYSTEM(CAL),!.

proces(6) if SYSTEM("Doclist"),!.
proces(7) if system("ComSh"),!,
proces(8).

NOTE(0) IF NL,
WRITE("   Press down arrow \25 to select an item or Esc to return to main menu."),!.

NOTE(1) IF nl,
WRITE(" Press down arrow \25 to select another item or press Enter to change "),
nl,
WRITE(" the drive and/or directory used to read input data and write output  "),
nl,
WRITE(" files. A global comment can be input to label any data files created."),
!.

NOTE(2) IF nl,
WRITE(" Press up or down arrow to select another item or press Enter to enter"),
nl,
WRITE(" the DOS shell. While under the DOS shell any DOS comain can be used  "),
nl,
WRITE(" just as under normal DOS. Type EXIT to return to NAIL when complete. "),
!.
```

```
NOTE(3) IF nl,
WRITE("  Press up or down arrow to select another item or press Enter to edit "),
nl,
WRITE("  the DeMENU step list. This list contains the measages and operations "),
nl,
WRITE("  acomplished when using NAIL. New procedures can be added using this. "),
!.

NOTE(4) IF nl,
WRITE("  Press up or down arrow to select another item or press Enter to change"),
nl,
WRITE("  the comments used to identify each data set in the selected Drv/Dir.  "),
nl,
WRITE("  The global comment is used to construct comments as data is created. "),
!.

NOTE(5) IF nl,
WRITE("  Press up or down arrow to select another item or press Enter to read or"),
nl,
WRITE("  to print the NAIL tutorial. This document is intended to give an over"),
nl,
WRITE("  view of how to use the NAIL package and is not a detailed write up. "),
!.

NOTE(6) IF NL,
WRITE("  Press up arrow \24 to select another item or press Enter to enter a"),
nl,
WRITE("  comment on the NAIL system to send to the Oak Ridge National "),
nl,
WRITE("  Laboratory. "),
!.
NOTE(_) IF NL,
WRITE("    Press up arrow \24 to select an item or Esc to return to main menu."),!.
```

[BLANK PAGE]

Appendix J.2

AddDir

```
Procedure AddDir(Var FileName: Path);

VAR
  HoldFileName: Path;
  I: Integer;

Begin  (* Procedure AddDir *)

  HoldFileName:=FileName;
  I:=Length(DrvDir);
  HoldFileName:=DrvDir + '\' + FileName;

  FileName:='';
  FileName:=HoldFileName;


End;  (* Procedure AddDir *)^Z
```

[BLANK PAGE]

Appendix J.3

AddFileExt

```
Procedure AddFileExt(Var FixFile:Path;Ext:String3);

VAR
  I,K: integer;
  HoldFileName: Path;


Begin  (* Procedure AddFileExt *)

   I:=1;
   HoldFileName:='';
   While ((FixFile[I] <> ' ') and (FixFile[I] <> '.')) Do I:=I + 1;
   HoldFileName:=FixFile;
   Delete(HoldFileName,I,64);
   HoldFileName:=HoldFileName + '.' + Ext;
   FixFile:='';
   FixFile:=HoldFileName;

End;  (* Procedure AddFileExt *)

^Z
```

[BLANK PAGE]

Appendix J.4

BoxWin

```
Procedure BoxWin(X1,Y1,X2,Y2:integer;Title:String50);

  ================================
    Box in defined Window
  ================================

Var
  X,Y:integer;

Begin  (* Procedure BoxWin *)

  Window(X1+1,Y1+1,X2-1,Y2-1);
  ClrScr;
  Window(1,1,80,25);

  (* Bottom of screen *)
  GotoXY(X1,Y2); Write(Chr(212));
  For X:=X1 + 1 to X2 - 1 Do Write(Chr(205));
  Write(Chr(190));

  (* Sides *)
  For Y:=Y1 to Y2 - 1 Do Begin
      GotoXY(X1,Y); Write(Chr(179));
      GotoXY(X2,Y); Write(Chr(179));
  End;


  If ((X2 = 80) and (Y2 = 24)) Then Begin
     GotoXY(X1,Y1);  InsLine;
  End;
  GotoXY(X1,Y1);  Write(Chr(213));
  Y:=(X2 - X1 + 1 - 2 - Length(Title)) DIV 2;
  For X:=X1 + 1 to Y + X1 Do Write(Chr(205));
  Write(Title);
  For X:=Y + 1 + Length(Title) + X1 to X2 -1 Do Write(Chr(205));
  Write(Chr(184));


  If ((X2 = 80) and (Y2 = 24)) Then Window(X1+1,Y1+1,X2-1,Y2)
  Else Window(X1 + 1,Y1 + 1,X2 - 1,Y2 - 1);

End; (* Procedure BoxWin *)
^Z
```

[BLANK PAGE]

Appendix J.5

DisplayDir

```
Procedure DisplayDir;


Begin

   GotoXY(32,6);  If TempDrv <> ' ' Then Write(TempDrv);
   GotoXY(37,8);  For I:=1 to Length(TempDir) Do Write(TempDir[I]);
End;

(*=====================================================================*)
Procedure DisplayCom;

VAR
  J,ComNo,I,NoChar,Row: Integer;
  StartN,EndN: Integer;
  Found: Boolean;

Begin  (* Procedure DisplayCom *)

   TmpCom1:=''; TmpCom2:=''; TmpCom3:=''; TmpCom4:=''; TmpCom5:='';
   (* Break up Comments *)
   NoChar:=Length(TempCom);
   Row:=11;
   EndN:=0; ComNo:=1;
   While (EndN + 1 <=NoChar) Do Begin
      StartN:=EndN + 1; Found:=False;
      While ((StartN <=NoChar) and (Found = False)) Do Begin
          If TempCom[StartN] = ' ' Then StartN:=StartN + 1
          Else Found:=True;
      End;
      EndN:=StartN + 50;
      If EndN > NoChar Then EndN:=NoChar;
      (* Break up into words *)
      Found:=False;
      If ((EndN <> NoChar) and (TempCom[EndN + 1] <> ' ')) Then
         While ((EndN > StartN) and (Found = False)) Do Begin
             If TempCom[EndN] <> ' ' Then
                EndN:=EndN - 1
             Else Found:=True;
         End;

      GotoXY(15,Row);
      For I:=StartN to EndN Do
         Write(TempCom[I]);
```

```
      Case ComNo of
         1: TmpCom1:=Copy(TempCom,StartN,EndN - StartN + 1);
         2: TmpCom2:=Copy(TempCom,StartN,EndN - StartN + 1);
         3: TmpCom3:=Copy(TempCom,StartN,EndN - StartN + 1);
         4: TmpCom4:=Copy(TempCom,StartN,EndN - StartN + 1);
         5: TmpCom5:=Copy(TempCom,StartN,EndN - StartN + 1);
      End;  (* Case Statement *)
      ComNo:=ComNo + 1;
      Row:=Row + 1;
   End;
   For I:=Length(TmpCom1) + 1 to 51 Do TmpCom1:=TmpCom1 + ' ';
   For I:=Length(TmpCom2) + 1 to 51 Do TmpCom2:=TmpCom2 + ' ';
   For I:=Length(TmpCom3) + 1 to 51 Do TmpCom3:=TmpCom3 + ' ';
   For I:=Length(TmpCom4) + 1 to 51 Do TmpCom4:=TmpCom4 + ' ';
   For I:=Length(TmpCom5) + 1 to 51 Do TmpCom5:=TmpCom5 + ' ';

End;  (* Procedure DisplayCom *)
^Z
```

Appendix J.6

Combine

```
Program Combine(input,output);
{$C-}
```

STEP 11 - Combine two files.This program is used for combining two data sets.

Deborah A. Holder

```
TYPE

    Str140 = string[140];
    HoldName = array [1..25] of char;
    RegRec = record
                ax,bx,cx,dx,bp,si,di,ds,es,flags: integer;
             end;
    Char12arr = array [1..12] of char;
    String255 = string[255];
    String50  = String[50];
    String20 = string[20];
    String10 = string[10];
    String6 = string[6];
    DPointer = ^DirRec;
    DirRec = record
                Size: real;
                Hr,Minu: integer;
                Year,Month,Day: integer;
                Next,Prev: DPointer;
                Name: String20;
             end;
    ImageType = array [1..4096] of char;
    Path = string[64];

VAR

    GFN,InFile1,InFile2,OutFile: Text;
    IFileName1,IFileName2,OFileName: Path;
    HeapTop: ^Integer;
    I,NoFiles: integer;
    Top,Bottom: DPointer;
```

```
    FirstRec,LastRec,NewRec: DPointer;
    Default,UppCase,Quit,ReqFile,ValFile: Boolean;
    Ans: Char;
    Field,MaxField: Integer;
    Buffer: Str140;
    Image: ImageType;
    CrtMode: Byte absolute $0040:$0049;
    MonoBuffer: ImageType absolute $B000:$0000;
    ColorBuffer: ImageType absolute $B800:$0000;
    ChkDir,ProgDrvDir,DrvDir: Path;
    TempCom: String255;
    TmpCom1,TmpCom2,TmpCom3,TmpCom4,TmpCom5: Path;
    HlpMsg1,HlpMsg2: String255;


{$IBoxWin.pas}
{$IHelp.pas}
{$IAddDir.pas}
{$IComments.pas}
{$IExists.pas}
{$IDiskSpace.pas}
{$IGetDir.pas}
{$IDirLst.pas}
{$IFileName.pas}
{$IReadField.pas}
{$IReadCom.pas}
{$IDisCom.pas}
{$IUpdateCom.pas}

Begin

    (* Get File to be Combined and open files *)
    GraphBackGround(Blue);
    TextBackGround(Blue);
    TextColor(Yellow);
    ClrScr;
    BoxWin(1,1,80,24,'  STEP 11 ');
    GotoXY(26,10);
    Write('>>> Combine Two Files <<<');
    GotoXY(1,1);
    GetDir(0,ProgdrvDir);
    Assign(GFN,'GFN_Nail');
    Reset(GFN);
    ChkDir:='';
    Readln(GFN,DrvDir);
    Close(GFN);
    ChkDir:=DrvDir;
```

```
If Length(ChkDir)= 2 Then ChkDir:=ChkDir + '\';
Delay(1000);
Mark(HeapTop);
GetDirectory('????????.D??');
ValFile:=False;
While ValFile = False Do Begin
   IFileName1:='                    ';
   ReqFile:=True;
   FileName(IFileName1,' First  ');
   If ProgDrvDir <> DrvDir Then AddDir(IFileName1);
   ValFile:=Exists(IFileName1);
   If ValFile = False Then Begin
      GotoXY(1,12); ClrEOL;
      GotoXY(15,12); Write(^G,'File Does Not Exist');
      Delay(3000);
   End;
End;
ValFile:=False;
While ValFile = False Do Begin
   IFileName2:='                    ';
   ReqFile:=True;
   FileName(IFileName2,' Second  ');
   If ProgDrvDir <> DrvDir Then AddDir(IFileName2);
   ValFile:=Exists(IFileName2);
   If ValFile = False Then Begin
      GotoXY(1,12); ClrEOL;
      GotoXY(15,12); Write(^G,'File Does Not Exist');
      Delay(3000);
   End;
End;

(* Get Output File and open File *)
ValFile:=True;
Repeat
   OFileName:='                    ';
   ReqFile:=True;
   FileName(OfileName,' Output  ');
   If ProgDrvDir <> DrvDir Then AddDir(OFileName);
   ValFile:=Exists(OFileName);
   If ValFile = True Then Begin
      GotoXY(1,12); ClrEOL;
      GotoXY(1,13); ClrEOL;
      GotoXY(17,12); Write(^G,'File Already Exist !!');
      GotoXY(17,13); Write('Do you wish to overwrite ?: _');
      GotoXY(WhereX -1,WhereY);
      Read(kbd,Ans);
      Ans:=UpCase(Ans);
```

```
        GotoXY(WhereX,WhereY);
        Write(Ans);
        If Ans = 'Y' Then ValFile:=False;
      End;
    Until (ValFile = False);
    Assign(InFile1,IFileName1);
    ReSet(InFile1);
    Assign(InFile2,IFileName2);
    ReSet(InFile2);
    Assign(OutFile,OFileName);
    ReWrite(OutFile);

    Release(HeapTop);
    ClrScr;
    GotoXY(15,10);
    Write('COMBINING FILES ');
    I:=1;
    While ((IFileName1[I] <> ' ') and (I <= 25)) Do Begin
      Write(UpCase(IFileName1[I]));
      I:= I + 1;
    End;
    Write(' and ');
    I:=1;
    While ((IFileName2[I] <> ' ') and (I <= 25)) Do Begin
      Write(UpCase(IFileName2[I]));
      I:= I + 1;
    End;
    GotoXY(15,12);
    Write('THE COMBINED FILES WILL BE WRITTEN TO ');
    For I:=1 to 25 Do Write(UpCase(OFileName[I]));
    While Not (EOF(InFile1)) Do Begin
      Readln(InFile1,Buffer);
      Writeln(OutFile,Buffer);
    End;
    Close(InFile1);

    While Not (EOF(InFile2)) Do Begin
      Readln(InFile2,Buffer);
      Writeln(OutFile,Buffer);
    End;
    Close(InFile2);
    Flush(OutFile);
    Close(OutFile);
    Default:=False;
    TempCom:='COMBINATION OF FILES ' + IFileName1 + ' AND ' + IFileName2;
    UpDateCom;

End.^Z
```

Appendix J.7

Comments

```
Procedure Comments(Check:String20);

TYPE
   BigStr = String[255];

VAR
  CFile: Text;
  CName: Path;
  CH: Char;
  Found: Boolean;
  Row,I: Integer;
  NoChar,StartN,EndN: Integer;
  Image: ImageType;
  Name: String20;
  TempCom,GenComments,DirCom,UserCom: BigStr;

Begin  (* Procedure Comments *)

   Name:='';
   Name:=Copy(Check,1,Length(Check) - 1);
   If CrtMode = 7 Then
      Image:=MonoBuffer
   Else
      Image:=ColorBuffer;
   BoxWin(10,10,70,20,' Press Space Bar to Exit ');
   TextColor(Black);
   TextBackGround(LightGray);
   ClrScr;
   CName:='DataList.cmt';
   If DrvDir <> ProgDrvDir Then AddDir(CName);
   Assign(CFile,CName);
   Reset(CFile);
   Found:=False;
   For I:=1 to Length(Name) Do Name[I]:=Upcase(Name[I]);
   Readln(CFile,GenComments);
   While ((NOT EOF(CFile)) and (Found = False)) Do Begin
      Readln(CFile,DirCom);
      For I:=1 to Length(DirCom) Do DirCom[I]:=UpCase(DirCom[I]);
      If (Pos(Name,DirCom)) <> 0 Then Found:=True;
      Readln(CFile,UserCom);
   End;
   Close(CFile);
   GotoXY(2,1);
```

```
   Writeln(Name);
   Row:=3;  TempCom:='';
   If Found = True Then TempCom:=UserCom
   Else TempCom:=GenComments;
   (* Break up Comments *)
   NoChar:=Length(TempCom);
   EndN:=0;
   While (EndN + 1 <=NoChar) Do Begin
      StartN:=EndN + 1; Found:=False;
      While ((StartN <=NoChar) and (Found = False)) Do Begin
          If TempCom[StartN] = ' ' Then StartN:=StartN + 1
          Else Found:=True;
      End;
      EndN:=StartN + 51;
      If EndN > NoChar Then EndN:=NoChar;
      (* Break up into words *)
      Found:=False;
      If ((EndN <> NoChar) and (TempCom[EndN + 1] <> ' ')) Then
         While ((EndN > StartN) and (Found = False)) Do Begin
             If TempCom[EndN] <> ' ' Then
                EndN:=EndN - 1
             Else Found:=True;
         End;

      GotoXY(2,Row);
      For I:=StartN to EndN Do Write(TempCom[I]);
      Row:=Row + 1;
   End;
   GotoXY(2,9);
   Read(Kbd,ch);
   If CrtMode = 7 Then
      MonoBuffer:=Image
   Else ColorBuffer:=Image;
   TextColor(Yellow);
   TextBackGround(Blue);
   Window(2,2,79,23);

End;  (* Procedure Comments *)
^Z
```

Appendix J.8

ComSh

```
Program ComSh(input,output);
```

Produces a user written comment sheet on NAIL. The comment sheet is sent to the printer and indicates where to mail the comments.

November 5, 1986
Deborah A. Holder

```
TYPE
   String50  = String[50];
   String255 = String[255];
   DateTimeType = String[8];
   RegRec = record
              ax,bx,cx,dx,bp,si,di,ds,es,flags: integer;
            end;
   String30  = String[30];
   Path      = String[64];
   ImageType = array [1..4096] of char;

VAR
  PrtDate: DateTimeType;
  Name: Path;
  Add1: Path;
  Add2: Path;
  Phone: Path;
  com1,com2,com3,com4,com5: Path;
  com6,com7,com8,com9,com10: Path;
  com11,com12,com13: Path;
  HlpMsg1,HlpMsg2: String255;
  UppCase,Quit: Boolean;
  Field,MaxField: Integer;
  Image: ImageType;
  CrtMode: Byte absolute $0040:$0049;
  MonoBuffer: ImageType absolute $B000:$0000;
  ColorBuffer: ImageType absolute $B800:$0000;
```

```
{$IBoxwin.pas}
{$IDate.pas}
{$IHelp.pas}
{$IReadField.pas}
(*==================================================================*)
Procedure ComScr;
(* Produces the Comment Screen *)

Var
  I: integer;

Begin  (* Procedure ComScr *)

  GotoXY(4,2);  Write('Name:  ______________________________________________  Date:  ',PrtDate);
  GotoXY(4,4);  Write('Address: ____________________________________________');
  GotoXY(4,5);  Write('  ____________________________________  Phone: ______________');
  GotoXY(4,7);  For I:=1 to 30 Do Write(Chr(205)); Write(' COMMENTS ');
  For I:=1 to 30 Do Write(Chr(205));
  GotoXY(7,9);  Write('______________________________________________________________');
  GotoXY(7,10); Write('______________________________________________________________');
  GotoXY(7,11); Write('______________________________________________________________');
  GotoXY(7,12); Write('______________________________________________________________');
  GotoXY(7,13); Write('______________________________________________________________');
  GotoXY(7,14); Write('______________________________________________________________');
  GotoXY(7,15); Write('______________________________________________________________');
  GotoXY(7,16); Write('______________________________________________________________');
  GotoXY(7,17); Write('______________________________________________________________');
  GotoXY(7,18); Write('______________________________________________________________');
  GotoXY(7,19); Write('______________________________________________________________');
  GotoXY(7,20); Write('______________________________________________________________');
  GotoXY(7,21); Write('______________________________________________________________');
  TextColor(Black);
  TextBackGround(LightGray);
  GotoXY(7,23); Write('          Press Esc Key to Print Comments and Exit          ');
  TextColor(Yellow);
  TextBackGround(Blue);

End;  (* Procedure ComScr *)
(*==================================================================*)
Procedure DoAddress;

Begin  (* Procedure DoAddress *)

   BoxWin(10,10,70,20,' Mail Comments To ');
   GotoXY(8,3);  Write('     R. C. DeLozier');
   GotoXY(8,4);  Write('     Oak Ridge National Laboratroy');
   GotoXY(8,5);  Write('     P.O. Box X   Bldg. 1000');
   GotoXY(8,6);  Write('     Oak Ridge, TN 37831 - 6333');

End;  (* Procedure DoAddress *)
```

```
(*==================================================================*)
Procedure PrtComments;

Begin  (* Procedure PrtComments *)

   Writeln(Lst,Chr(15));
   Writeln(Lst);
   Writeln(Lst);
   Writeln(Lst,' ':14,PrtDate);
   Writeln(Lst);
   Writeln(Lst,' ':14,'NAIL COMMENTS');
   Writeln(Lst);
   Writeln(Lst);
   Writeln(Lst);
   Writeln(Lst,' ':14,'To:     R. C. DeLozier');
   Writeln(Lst,' ':14,'        Oak Ridge National Laboratory');
   Writeln(Lst,' ':14,'        P.O. Box X  Bldg. 1000');
   Writeln(Lst,' ':14,'        Oak Ridge, TN 37831 - 6333');
   Writeln(Lst);
   Writeln(Lst);
   Writeln(Lst);
   Writeln(Lst);
   Writeln(Lst);
   Writeln(Lst,' ':34,'-------------------------- Comments --------------------------');
   Writeln(Lst);
   Writeln(Lst,' ':34,Com1);
   Writeln(Lst,' ':34,Com2);
   Writeln(Lst,' ':34,Com3);
   Writeln(Lst,' ':34,Com4);
   Writeln(Lst,' ':34,Com5);
   Writeln(Lst,' ':34,Com6);
   Writeln(Lst,' ':34,Com7);
   Writeln(Lst,' ':34,Com8);
   Writeln(Lst,' ':34,Com9);
   Writeln(Lst,' ':34,Com10);
   Writeln(Lst,' ':34,Com11);
   Writeln(Lst,' ':34,Com12);
   Writeln(Lst,' ':34,Com13);
   Writeln(Lst);
   Writeln(Lst);
   Writeln(Lst);
   Writeln(Lst,' ':14,'Name:    ',Name);
   Writeln(Lst,' ':14,'Address: ',Add1);
   Writeln(Lst,' ':14,'         ',Add2);
   Writeln(Lst,' ':14,'Phone:   ',Phone);
   Writeln(Lst);
   Writeln(Lst,^L);

End;  (* Procedure PrtComments *)
(*==================================================================*)
```

```
Begin  (* Program ComSh *)

  GraphBackGround(Blue);
  TextBackGround(Blue);
  ClrScr;
  BoxWin(1,1,80,24,' NAIL COMMENT SHEET ');
  PrtDate:=Date;
  ComScr;

  MaxField:=17;  Field:=1; Quit:=False; UppCase:=False;
  Name:='';  Add1:='';  Add2:='';  Phone:='';
  Com1:='';  Com2:='';  Com3:='';  Com4:='';
  Com5:='';  Com6:='';  Com7:='';  Com8:='';
  Com9:='';  Com10:='';  Com11:='';  Com12:='';  Com13:='';
  While ((Field <=MaxField) and (Quit = False)) Do
      Case Field of
          1: Begin
                 HlpMsg1:='                Enter Your Name';
                 HlpMsg2:='';
                 ReadField(Name,13,2,30);
             End;

          2: Begin
                 HlpMsg1:='                Enter Your Address';
                 HlpMsg2:='';
                 ReadField(Add1,13,4,30);
             End;

          3: Begin
                 HlpMsg1:='                Enter Your Address';
                 HlpMsg2:='';
                 ReadField(Add2,13,5,30);
             End;

          4: Begin
                 HlpMsg1:='              Enter Your Phone Number';
                 HlpMsg2:='';
                 ReadField(Phone,56,5,18);
             End;

          5: Begin
                 HlpMsg1:='              Enter Your Comments';
                 HlpMsg2:='';
                 ReadField(Com1,7,9,64);
             End;
```

```
6: Begin
      HlpMsg1:='            Enter Your Comments';
      HlpMsg2:='';
      ReadField(Com2,7,10,64);
   End;

7: Begin
      HlpMsg1:='            Enter Your Comments';
      HlpMsg2:='';
      ReadField(Com3,7,11,64);
   End;

8: Begin
      HlpMsg1:='            Enter Your Comments';
      HlpMsg2:='';
      ReadField(Com4,7,12,64);
   End;

9: Begin
      HlpMsg1:='            Enter Your Comments';
      HlpMsg2:='';
      ReadField(Com5,7,13,64);
   End;

10: Begin
      HlpMsg1:='            Enter Your Comments';
      HlpMsg2:='';
      ReadField(Com6,7,14,64);
   End;

11: Begin
      HlpMsg1:='            Enter Your Comments';
      HlpMsg2:='';
      ReadField(Com7,7,15,64);
   End;

12: Begin
      HlpMsg1:='            Enter Your Comments';
      HlpMsg2:='';
      ReadField(Com8,7,16,64);
   End;

13: Begin
      HlpMsg1:='            Enter Your Comments';
      HlpMsg2:='';
      ReadField(Com9,7,17,64);
   End;
```

```
        14: Begin
              HlpMsg1:='              Enter Your Comments';
              HlpMsg2:='';
              ReadField(Com10,7,18,64);
            End;

        15: Begin
              HlpMsg1:='              Enter Your Comments';
              HlpMsg2:='';
              ReadField(Com11,7,19,64);
            End;

        16: Begin
              HlpMsg1:='              Enter Your Comments';
              HlpMsg2:='';
              ReadField(Com12,7,20,64);
            End;

        17: Begin
              HlpMsg1:='              Enter Your Comments';
              HlpMsg2:='';
              ReadField(Com13,7,21,64);
            End;

      End;  (* Case Statement *)

  DoAddress;
  PrtComments;

End.   (* Program ComSh *)^Z
```

Appendix J.9

Data_Dir

```
Program Data_Dir(Input,output);
{$C-}

TYPE
   String10  = String[10];
   String50  = String[50];
   Path      = String[64];
   String255 = String[255];
   ImageType = array [1..4096] of char;

VAR
   Default,Comments: Text;
   GFN: Text;
   PN,I: Integer;
   Field,MaxField: Integer;
   TFile,Line,TempCom: String255;
   TmpCom1,TmpCom2,TmpCom3,TmpCom4,TmpCom5: Path;
   TempDrv: Char;
   ProgDrvDir,TempDir: Path;
   ComDir,ComFile: Path;
   HlpMsg1,HlpMsg2: String255;
   Par: Path;
   UppCase,DirOK,Quit: Boolean;
   Image: ImageType;
   CrtMode: Byte absolute $0040:$0049;
   MonoBuffer: ImageType absolute $B000:$0000;
   ColorBuffer: ImageType absolute $B800:$0000;

{$IBoxwin.pas}
{$IHelp.pas}
{$IExists.pas}
{$IValDir.pas}
{$IBreak.pas}
{$IReadField.pas}
(*=================================================================*)
  Procedure DirScr;

  Begin  (* Procedure DirScr *)

     GotoXY(25,2);  Write('Default Drive and/or Directory');
     GotoXY(30,3);  Write('for Data input/output');

     GotoXY(5,6);   Write('Default Drive  (e.g. A ) : _');
     GotoXY(5,8);   Write('Default Directory  (e.g. \P3) : ______________________________________');
```

```
    TextBackGround(LightGray);
    TextColor(Black);
    GotoXY(5,23);  Write(' F1: Help                                                    ');
    TextBackGround(Blue);
    TextColor(Yellow);

  End;  (* Procedure DirScr *)
(*=================================================================*)

Procedure ComScr;

  Begin  (* Procedure ComScr *)

    GotoXY(5,11);  Write('Comments: ______________________________________________');
    GotoXY(5,12);  Write('          ______________________________________________');
    GotoXY(5,13);  Write('          ______________________________________________');
    GotoXY(5,14);  Write('          ______________________________________________');
    GotoXY(5,15);  Write('          ______________________________________________');
    GotoXY(1,18);  ClrEOL;
    TextBackGround(LightGray);
    TextColor(Black);
    GotoXY(5,23);  Write(' F1: Help                          Esc Key: EXIT to NAIL     ');
    TextBackGround(Blue);
    TextColor(Yellow);

  End;  (* Procedure ComScr *)
(*=================================================================*)
  Procedure ReadDir;

   VAR
      SendArr: Path;

  Begin  (* Procedure ReadDir *)

    Quit:=False; Field:=1; MaxField:=2;  UppCase:=true;
    While Field <= MaxField Do Begin
       SendArr:='';
       Case Field of
          1: Begin
               SendArr:=SendArr + TempDrv;
               HlpMsg1:='Enter the DRIVE on which the data sets and reports reside. (e.g. A:)';
               HlpMsg2:='';
               ReadField(SendArr,32,6,1);
               If Length(SendArr) <> 0 Then
                  TempDrv:=SendArr[1]
               Else TempDrv:=' ';
               GotoXY(5,19);
```

```
            If TempDrv = ' ' Then Begin
               Write(^G,'This field is required.  Must be a valid Drive.');
               Field:=1;
               Quit:=False;
            End
            Else ClrEOL;
          End;
       2: Begin
            For I:=1 to 26 Do SendArr:=SendArr + ' ';
            For I:=1 to Length(TempDir) Do SendArr[I]:=TempDir[I];
            HlpMsg1:='Enter the DIRECTORY on which the data sets and reports ';
            HlpMsg2:='reside. (e.g. \DirName)';
            ReadField(SendArr,37,8,26);
            TempDir:=Copy(SendArr,1,26);
          End;
      End;  (* Case Statement *)
    End;  (* While Loop *)
  End;   (* Procedure ReadDir *)
(*=====================================================================*)
  Procedure ReadCom;

   VAR
     SendArr: Path;

  Begin  (* Procedure ReadCom *)

    Quit:=False; Field:=1; MaxField:=5; UppCase:=True;
    HlpMsg1:='General Comments concerning this session in NAIL.';
    HlpMsg2:='Press the Esc key to continue to NAIL.';
    While ((Field <= MaxField) and (Quit = False)) Do Begin
       SendArr:='';
       Case Field of
          1: Begin
               For I:=1 to 51 Do SendArr:=SendArr + ' ';
               For I:=1 to Length(TmpCom1) Do SendArr[I]:=TmpCom1[I];
               ReadField(SendArr,15,11,51);
               TmpCom1:=Copy(SendArr,1,51);
             End;
          2: Begin
               For I:=1 to 51 Do SendArr:=SendArr + ' ';
               For I:=1 to Length(TmpCom2) Do SendArr[I]:=TmpCom2[I];
               ReadField(SendArr,15,12,51);
               TmpCom2:=Copy(SendArr,1,51);
             End;
```

```
        3: Begin
             For I:=1 to 51 Do SendArr:=SendArr + ' ';
             For I:=1 to Length(TmpCom3) Do SendArr[I]:=TmpCom3[I];
             ReadField(SendArr,15,13,51);
             TmpCom3:=Copy(SendArr,1,51);
           End;
        4: Begin
             For I:=1 to 51 Do SendArr:=SendArr + ' ';
             For I:=1 to Length(TmpCom4) Do SendArr[I]:=TmpCom4[I];
             ReadField(SendArr,15,14,51);
             TmpCom4:=Copy(SendArr,1,51);
           End;
        5: Begin
             For I:=1 to 51 Do SendArr:=SendArr + ' ';
             For I:=1 to Length(TmpCom5) Do SendArr[I]:=TmpCom5[I];
             ReadField(SendArr,15,15,51);
             TmpCom5:=Copy(SendArr,1,51);
           End;
      End;  (* CaseStatement *)

  End;   (* While Loop *)
End;    (* Procedure ReadCom *)

(*=================================================================*)
Procedure BuildCom;

(*=================================================================*)
   Procedure FindEnd(VAR HoldStr:Path);

    VAR
      I: integer;

   Begin (* Procedure FindEnd *)

      I:=Length(HoldStr);
      While HoldStr[I] = ' ' Do I:=I - 1;
      Delete(HoldStr,I + 1,51 - I);

   End;  (* Procedure FindEnd *)
(*=================================================================*)

Begin  (* Procedure BuildCom *)

   If Length(TmpCom1) > 1 Then FindEnd(TmpCom1);
   If Length(TmpCom2) > 1 Then FindEnd(TmpCom2);
   If Length(TmpCom3) > 1 Then FindEnd(TmpCom3);
```

```
      If Length(TmpCom4) > 1 Then FindEnd(TmpCom4);
      If Length(TmpCom5) > 1 Then FindEnd(TmpCom5);
      TempCom:='';
      TempCom:=TmpCom1 + ' ' + TmpCom2 + ' ' + TmpCom3 + ' ' + TmpCom4 + ' ' + TmpCom5;

  End;  (* Procedure BuildCom *)
(*=====================================================================*)
  Procedure BuildCmt;

  VAR
    Temp: Text;
    BackUp: Text;
    BackFile: Path;
    TempFile: Path;

  Begin  (* Procedure BuildCmt *)

    TempFile:='Temp.zzz';
    TempFile:=ComDir + '\' + TempFile;
    BackFile:='DataList.bck';
    BackFile:=ComDir + '\' + BackFile;
    Assign(Temp,TempFile);
    Rewrite(Temp);
    Reset(Comments);
    If NOT EOF(Comments) Then Readln(Comments,Line);
    Writeln(Temp,TempCom);
    While NOT EOF(Comments) Do Begin
       Readln(Comments,Line);
       Writeln(Temp,Line);
    End;
    Close(Temp);
    Close(Comments);
    If Par <> '**' Then Erase(Comments)
    Else Begin
       If Exists(BackFile) Then Begin
          Assign(BackUp,BackFile);
          Erase(BackUp);
       End;
       ReName(Comments,BackFile);
    End;

    Rename(Temp,ComFile);

  End;  (* Procedure BuildCmt *)
(*=====================================================================*)
```

```
  Procedure BuildGFN;

  VAR
    I: integer;
    WrtDrvDir: Path;

  Begin   (* Procedure BuildGFN *)

    Assign(GFN,'GFN_NAIL');
    Rewrite(GFN);
    WrtDrvDir:='';
    I:=Length(TempDir);
    While TempDir[I] = ' ' Do Begin
       Delete(TempDir,I,1);
       I:=I - 1;
    End;
    I:=Length(TempDir);
    If I <> 0 Then Begin
       If (TempDir[I] = '\') Then Delete(TempDir,I,1);
    End;

    WrtDrvDir:=TempDrv + ':' + TempDir;
    Writeln(GFN,WrtDrvDir);
    I:=Length(WrtDrvDir);
    ComDir:=WrtDrvDir;
    Writeln(GFN,'**');
    Close(GFN);

  End;   (* Procedure BuildGFN *)
(*=================================================================*)

Begin

  GraphBackGround(Blue);
  TextBackGround(Blue);
  TextColor(Yellow);
  ClrScr;
  BoxWin(1,1,80,24,'   NAIL   ');
  DirScr;
  Pn:=Paramcount;
  Par:='';
  If Pn > 0 Then Par:=Paramstr(1);

  TempCom:=''; ProgDrvDir:=''; TFile:=''; Line:='';
  GetDir(0,ProgDrvDir);
  For I:=1 to Length(ProgDrvDir) Do ProgDrvDir[I]:=UpCase(ProgDrvDir[I]);
  If Par <> '**' Then Begin
```

```
  If Exists('GFN_NAIL') = True Then Begin
     Assign(GFN,'GFN_Nail');
     Reset(GFN);
     Readln(GFN,TempCom);
     Readln(GFN,TFile);
     Close(GFN);
  End;
End
Else Begin
   If Pn > 1 Then TempCom:=Paramstr(2)
   Else TempCom:=ProgDrvDir;
End;
For I:=1 to Length(TempCom) Do TempCom[I]:=UpCase(TempCom[I]);
TempDrv:=TempCom[1];
I:=Pos(':',TempCom);
If I <> 0 Then
  TempDir:=Copy(TempCom,I+1,64)
Else TempDir:=' ';
For I:=Length(TempDir) + 1 to 26 Do TempDir[I]:=' ';
DirOK:=False;
Repeat
  DisplayDir;
  ReadDir;
  I:=Length(TempDir);
  While TempDir[I] = ' ' Do Begin
     Delete(TempDir,I,1);
     I:=I - 1;
  End;
  If Length(TempDir) = 0 Then
     DirOK:=ValDir(TempDrv + ':' + '\')
  Else DirOK:=ValDir(TempDrv + ':' + TempDir);
  If NOT DirOK Then Begin
     GotoXY(5,18);  Write(^G,'INVALID DRIVE AND/OR DIRECTORY');
  End;
Until (DirOK = True);
BuildGFN;
ComFile:='DataList.cmt';
ComFile:=ComDir + '\' + ComFile;
Assign(Comments,ComFile);
If Exists(ComFile) = True Then Begin
  Reset(Comments);
  Readln(Comments,Line);
End
Else Begin
  Rewrite(Comments);
  Line:='';
```

```
  End;
  TempCom:='';
  TempCom:=Line;
  ComScr;
  DisplayCom;
  ReadCom;
  BuildCom;
  BuildCmt;

End.^Z
```

Appendix J.10

Date

```
(*==========================================================================*)
 Function Date:DateTimeType;

  (*----------------------------*)
  (* Get Current Date from DOS *)
  (*----------------------------*)

  Var
    I:integer;
    Reg: RegRec;
    m,d,y,w: DateTimeType;

   Begin (* Function Date *)

     Reg.ax:=$2A00;
     intr($21,reg);
     str(Reg.cx:4,y);
     delete(y,1,2);
     str(hi(reg.dx):2,m);
     str(lo(reg.dx):2,d);
     w:=m + '/' + d + '/' + y;
     For I:= 1 to Length(w) do if w[i] = ' ' then w[I]:='0';
     Date:=w;

   End; (* Function Date *)
^Z
```

[BLANK PAGE]

Appendix J.11

DeEditor

DeEditor

by
R. C.DeLozier

```
program ReWord;
const           strxx =240;
type
    str80        = string[strxx]; {controls text length of two lines}

    str255       = string[255];
    str20        = string[28];
    linep        = ^line;
    line         = record
                     text1   :str80;
                     text2   :str80;
                     num     :integer;
                     next    :linep;
                     prior   :linep;
                     end;
    dataitem      = line;
    filetype      = text;
var
    text,fildrv    : filetype;
    start,last     : linep;
    done           : boolean;
    fname,efile,dfile,drvdir
                   : str20;
    ch,c1,c2,o,
    yyy,xxx,x,y
                   : integer;
procedure beep(f:integer);
begin
      sound(f);
      delay(250);
      nosound;
end;

procedure top(x,y:integer);
```

```
begin
        if (x<1) or (x>80) then x:=80;
        window(1,1,80,2);gotoxy(x,y);
        if (x<1) or (x>80) then writeln;
        textbackground(7);
        textcolor(0);
end;
procedure bottom(var x,y:integer);
begin
    if (x<1) or (x>80) then x:=1;
    if y<1 then y:=1;
    if y>25 then y:=25;
    window(1,3,80,25);
    textbackground(0);
    textcolor(14);
    gotoxy(x,y);
end;
function chars(var c1,c2:integer):char;
var
    ch          :char;
begin
    c1:=0;c2:=0;
    read(kbd,ch);c1:=ord(ch);chars:=ch;
    if (c1=27) and keypressed then
    begin
        read(kbd,ch);
        c2:=ord(ch);
    end;

end; {End of chars}
function menu(var o:integer): char;
type li = array[1..8] of string[50];
const
     ti:li=('1.Edit  an existing line ','2.Insert a new line of text  ',
     '3.Delete an existing line ','4.Move  a line to another location ',
     '5.Copy  a line to another location ',
     '6.List  current file   ','7.Save   file with current changes ',
     '8.Quit  ');
var
    ch          :char;
    vch         :string[1];
    zz          :string[8];
    c1,c2,i,x1,y1
                :integer;
begin
    x1:=wherex;y1:=wherey;
    c1:=1;c2:=2;
```

```
    top(c1,c2);
    repeat
    ClrScr;
    for i:=1 to 8 do
    begin
         textbackground(7);
         textcolor(0);
         if i=o then
         begin
              textbackground(0);
              textcolor(31);
         end;  gotoxy(2+(i-1)*10,1);
         zz:=ti[i];
         write(zz);
         textbackground(7);
         textcolor(0);
    end;  yyy:=wherey+1;
          gotoxy(2,yyy);
          write(ti[o]);
          xxx:=wherex;
          gotoxy(xxx+2,yyy);
          if o=8 then
           begin
             write('( will ');
             textcolor(16);
             write('NOT');
             textcolor(0);write(' automaticity save changes )');
           end;
          write(' Enter choice: ');
          ch:=chars(c1,c2); ch:=upcase(ch);
          if c1=13 then
                        begin
                        str(o:1,vch);
                        ch:=vch;
                        end;
          if c2=77 then o:=o+1;
          if c2=75 then o:=o-1;
          if o<1 then o:=8;
          if o>8 then o:=1;
    until (ch>='1') and (ch<='Z');
    writeln;writeln;
    bottom(x1,y1);

    menu:=ch;
end;  {End of menu}
function find(lnum:integer):linep;
```

```
var
   i:linep;
begin
   i:=start;
   find:=nil;
   while i<>nil do begin
         if lnum=i^.num then find:=i;
         i:=i^.next;
   end;
end;  {End of find}
procedure patch(lnum,incr:integer);
var
   i:linep;
begin
   i:=find(lnum);
   while (i<>nil) do begin
         i^.num:=i^.num+incr;
         i:=i^.next;
   end;
end;  {End of patch}
function StoreS(info,start:linep;var last:linep):linep;
var
   old,top          :^line;
   done             :boolean;
begin
   top:=start;
   old:=nil;
   done:=false;
   if start=nil then
   begin
         info^.next:=nil;
         last:=info;
         info^.prior:=nil;
         StoreS:=info;
   end else
   begin
         while (start<>nil) and (not done) do
         begin
               if start^.num < info^.num then
               begin
                     old:=start;
                     start:=start^.next;
               end else
               begin
                     if old<>nil then
                     begin
```

```
                    old^.next:=info;
                    info^.next:=start;
                    start^.prior:=info;
                    info^.prior:=old;
                    StoreS:=top;
                    done:=true;
                end else
                begin
                    info^.next:=start;
                    info^.prior:=nil;
                    StoreS:=info;
                    done:=true;
                end;
            end;
        end;
        if not done then
            begin
                last^.next:=info;
                info^.next:=nil;
                info^.prior:=last;
                last:=info;
                StoreS:=top;
            end;
        end;
end;  {End of StoreS}
function Delet(start:linep;key:integer):linep;
var
   temp,temp2      :linep;
   done            :boolean;

begin
   if start^.num=key then
   begin
        Delet:=start^.next;

        if start^.next <> nil then
        begin
             temp:=start^.next;
             temp^.prior:=nil;
        end;
        dispose(start);       patch(key+1,-1);
   end else
   begin
        done:=false;
        temp:=start^.next;
        temp2:=start;
        while (temp<>nil) and (not done) do
        begin
```

```
            if temp^.num=key then
            begin
                temp2^.next:=temp^.next;
                if temp^.next<>nil then
                temp^.next^.prior:=temp2;
                done:=true;
                last:=temp^.prior;
                dispose(temp);
            end else
            begin
                temp2:=temp;
                temp:=temp^.next;
            end;
        end;
        delet:=start;
        if not done then Writeln(' Not found!')
        else patch(key+1,-1);
    end;
end; {End of Delet}
procedure remove;
var
   rem          :linep;
   ch           :char;
   num          :integer;
   x,y,c1,c2    :integer;
begin
   x:=wherex;y:=wherey;top(1,2);
   write(' Enter line to delete: ');
   read(num); c1:=0;
   bottom(x,y);
   rem:=find(num);textcolor(31);
   write(rem^.text1);   writeln(rem^.text2);     textcolor(14);
   x:=wherex;y:=wherey;top(81,2);
   write(' Delete this line (Y/N)? ');
   repeat ch:=chars(c1,c2); until c1>32;
   if (ch='Y') or (ch='y') then start:=delet(start,num);
   bottom(x,y);
end; {End of remove}

procedure change(var Line,Line2:str80;x,y:integer);
var
ch,CCX          :char;
cc,dd,bb        :str80;
i,c1,c2,c3,p,di,
yy,xx,LL,LL2,P2,
STRX,DL,LT,lcc,
ldd,kk,y2,x2
                :integer;
```

```
begin
     STRX:=strxx;
     p:=1;di:=1;
     gotoxy(x,y);
     c1:=0;c2:=0;c3:=0;dd:='';
     LL2:=LENGTH(LINE2); LL:=length(line);
     LT:=LL+LL2; yy:=trunc(lt/80);
     if (y+yy)>16 then
     begin y:=y-6;writeln;writeln;writeln;writeln;writeln;writeln;end;
     while c3<>27 do
     begin
       gotoxy(x,y);
       LL2:=LENGTH(LINE2); LL:=length(line);
       IF (LL<STRX) and (ll2>0) THEN
          BEGIN
               DL:=STRX-LL;
               LINE:=LINE+copy(LINE2,1,dl);
               LINE2:=COPY(LINE2,DL+1,LL2);
          END;
       LL2:=LENGTH(LINE2); LL:=length(line);
       LT:=LL+LL2;P2:=P-LL;ldd:=p-lt;dd:='';
       y2:=y+trunc(ll/80);x2:=x+ll;x2:=(x2 mod 80);
       if (p>lt+1) then  for kk:=2 to ldd do dd:=dd+' ';
       textcolor(15);textbackground(1);
       if p<(LL+2) then write(line,LINE2) else
                                        begin
                                             gotoxy(x2,y2);
                                             write(line2);
                                        end;
       textbackground(0);textcolor(14);write(' ');
       top(1,1);
       if lt=0 then write(' Esc to stop edit ')
               else write(' Esc to enter text') ;
       textcolor(16);
       if di=0 then write(' Overwrite');
       if di=1 then write(' Insert   ');
       textcolor(0);
       write('       Line length: ',LT:3,
       ' ASCII [',c1:3,',',c2:3,'] Cursor: ',p:3);
       xx:=x+p-2;
       yy:=trunc(xx/80);yy:= y+yy;
       xx:=(xx mod 80)+1;
       bottom(xx,yy);
       repeat ch:=chars(c1,c2); until (c1=27) or (c1>31) or (c1=8);
       if c1<>13 then cc:=dd+ch;lcc:=length(cc);
       if (di=1) then
```

```
        begin
        if (c2=0) and (c1<>8) and (c1<>27) then
                        begin
                         if p>lt then line2:=line2+cc
                         else
                           begin
                              IF (LL+lcc>=STRX) and (p2<1) THEN
                                      BEGIN
                                       CCX:=COPY(LINE,LL,lcc);
                                       INSERT(CCX,LINE2,1);
                                      END ;
                              IF P2<1 THEN
                                      insert(cc,line,p)
                                   ELSE
                                      INSERT(CC,LINE2,P2);
                           end;
                         p:=p+1;
                        end;
        end else
        if (c2=0) and (c1<>8) and (c1<>27) then
                         begin
                          IF P2<0 THEN bb:=copy(line,p+1,ll)
                                  ELSE BB:=COPY(LINE2,P2+1,LL2);
                          IF P2<1 THEN line:=copy(line,1,p-1)+cc+bb
                                  ELSE line2:=copy(line2,1,p2-1)+cc+bb;
                          p:=p+1;
                         end;
  if (c1=27) and (c2=0) then c3:=27;
  if c2=82 then di:=(di+1) mod 2;

  if (c2=83) then  IF (P2<1) THEN delete(line,p,1)
                         ELSE DELETE(LINE2,P2,1);
  if (c1=8) and (p<=lT+1) and (p>1) then  IF (P2<2) THEN
                                           delete(line,p-1,1)
                                           ELSE DELETE(LINE2,P2-1,1);

  if (c2=75) or ((c1=8)and (p>1) and (p<=lt+1)) then p:=p-1;
  if c2=77 then p:=p+1;
  if (c2=71) and (p>1) then p:=1;
  if c2=79 then p:=lt+1;
  if c2=72 then p:=p-80;
  if c2=80 then p:=p+80;
  if (p>strx+ll) then begin p:=strx+ll;beep(2000);end;
  if (p<1) then begin p:=1;beep(1000);end;
end;
```

```
c1:=0;c2:=0;
writeln;
end; {end of change}
procedure editLine;
var
   rem             :linep;
   ch              :char;
   num             :integer;
   x,y,c1,c2       :integer;
   line1,line2     :str80;
begin
   writeln;
   x:=wherex;y:=wherey;top(1,2);
   write(' Enter line to edit: ');
   read(num); c1:=0;
   bottom(x,y);
   rem:=find(num);textcolor(0);textbackground(7);
   line1:=rem^.text1;
   line2:=rem^.text2;
   change(line1,line2,x,y);writeln;writeln;writeln;writeln;writeln;writeln;
   x:=wherex;y:=wherey;top(81,2);
   write(' Replace with this line (Y/N)? ');
   repeat ch:=chars(c1,c2); until c1>32;
   if (ch='Y') or (ch='y') then
       begin
        rem^.text1:=line1;
        rem^.text2:=line2;
       end;
   bottom(x,y);
end; {End of editLine}

procedure enter(var x,y:integer);
{<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<}
var
   info,xx         :linep;
   num ,vx,vy,c1,c2,strL1
                      :integer;
   line,line2      :str80;
   ch              :char;
   done            :boolean;
begin
     strL1:=strxx;
   done:=false;
   vx:=1;vy:=2;
   top(vx,vy);
   write(' Enter starting line number: ');
   read(num);
```

```
   repeat
         top(vx,vy);  write(' Entering line:',num:3,'              ');
         bottom(x,y);
         new(info);
         info^.num:=num;
         x:=wherex;y:=wherey;
         xx:=find(num);        line:='';line2:='';
         change(line,line2,x,y);
         writeln;writeln;writeln;writeln;writeln;writeln;
         x:=wherex;y:=wherey;
         info^.text1:=line;info^.text2:=line2;
         if (length(info^.text1)=0) then done:=true
         else begin
               if xx<>nil then patch(num,1);
               start:=StoreS(info,start,last);
         end;
         num:=num+1;writeln;y:=wherey;
   until done;
end;  {End of Enter}
procedure Display(start:linep);
label outa,outb;
var
   im,c1,c2
         :integer;
   ch    :char;
begin
      textbackground(1);
      while start<>nil do
      begin
            im:= start^.num;
            write(im,': ');
            write(start^.text1);          writeln(start^.text2);
            start:=start^.next;
            if im mod 4 =0 then
                              begin
                                    write(' More - Esc to stop any key to continue');
                                    repeat ch:=chars(c1,c2);
                                    if (c1=27) and (c2=0) then goto outb;
                                    if  c1>0 then goto outa;
                                    until keypressed;
                              outa: writeln;
                              end;

      end;
      write(' Hit any key to continue');read(kbd,ch);
      outb:c1:=0;c2:=0;
      writeln;
```

```
end;  {End of Display}
{>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>}
procedure save(var f:filetype;fn:str20;start:linep);
begin
     writeln(' Saving  file ',fn);
     rewrite(f);
     while start<>nil do
     begin
          writeln(f,start^.text1,start^.text2);
          start:=start^.next;
     end;
close(f);
end;  {End of save}
function load(var f:filetype;fn:str80):linep;
var
   temp            :linep;
   ffv             :filetype;
   ff              :str20;
   s1,s2           :str80;
   ip,iL,pp,cx     :integer;
begin
   iL:=length(fn);ip:=iL;pp:=pos('.',fn);
   if pp>0 then ip:=pp-1;
   ff:=copy(fn,1,ip)+'.bak';
   assign(ffv,ff);
   writeln(' Loading file ',fn);
   reset(f);
   while start<>nil do
   begin
        temp:=start^.next;
        dispose(start);
        start:=temp;
   end;
   cx:=1;
   start:=nil;last:=nil;
   while not eof(f) do
   begin
        new(temp);
        readln(f,s1,s2); temp^.num:=cx;cx:=cx+1;
        temp^.text1:=s1;temp^.text2:=s2;
        start:=stores(temp,start,last);
   end;
   load:=start;
   save(ffv,ff,start);
end;  {End of load}
procedure moveLine(var x,y:integer);
```

```
var
   info,exl       :linep;
   num,c1,c2,num2
                  :integer;
   ch             :char;
begin
      top(81,2);
      write(' Enter line number to move: ');
      read(num);
      exl:=find(num);
      bottom(x,y);
      writeln(exl^.text1,exl^.text2);
      x:=wherex;y:=wherey;
      top(81,2);
      write(' Is this line (Y/N) ');
      repeat ch:=chars(c1,c2) until (c1>32);
      if (ch='y') or (ch='Y') then
      begin
            writeln;write(' Enter distination line number: ');
            read(num2);
            new(info);
            info^.num:=num2;info^.text1:=exl^.text1;info^.text2:=exl^.text2;
            if find(num2)<>nil then patch(num2,1);
            start:=stores(info,start,last);
            start:=delet(start,num);
      end;
      bottom(x,y);
end;
procedure copyLine(var x,y:integer);
var
   info,exl   :linep;
   num,c1,c2,num2
               :integer;
   ch          :char;
begin
      top(81,2);
      write(' Enter line number to copy: ');
      read(num);
      exl:=find(num);
      bottom(x,y);
      writeln(exl^.text1,exl^.text2);
      x:=wherex;y:=wherey;
      top(81,2);
      write(' Is this line (Y/N) ');
      repeat ch:=chars(c1,c2) until (c1>32);
      if (ch='y') or (ch='Y') then
      begin
```

```
            writeln;write(' Enter distination line number: ');
            read(num2);
            new(info);
            info^.num:=num2; info^.text1:=ex1^.text1;info^.text2:=ex1^.text2;
            if find(num2)<>nil then patch(num2,1);
            start:=stores(info,start,last);
        end;
        bottom(x,y);
end;
begin  {start of reword}
        textcolor(15);textbackground(1);
        clrscr;gotoxy(35,1);write    ('NAIL');
                gotoxy(33,2);writeln('DeEditor');
        o:=1;
        dfile:='gfn_nail';
        assign(fildrv,dfile);
        reset(fildrv);
        readln(fildrv,drvdir);
        close(fildrv);
        if drvdir=' ' then drvdir:='' else drvdir:=drvdir+'\';
        if paramcount>0 then fname:=drvdir+paramstr(1) else  fname:='QDeLogic';;
        start:=nil;
        last:=nil;
        done:=false;
        assign(text,fname);assign(text,fname);
      {$I-} reset(text) {$I+};
      if not (ioresult=0) then rewrite(text) else start:=load(text,fname);

        repeat
                x:=wherex;y:=wherey;
                if y>24 then
                        begin
                                writeln;
                                y:=y-1;
                        end;
                case menu(o) of
                        '1' :editLine;
                        '2' :enter(x,y);
                        '3' :remove;
                        '4' :moveLine(x,y);
                        '5' :copyLine(x,y);
                        '6' :display(start);
                        '7' :save(text,fname,start);
                        '8' :done:=true;
                        'M' :moveLine(x,y);
                        'C' :copyLine(x,y);
                        'I' :enter(x,y);
```

```
                    'D' :remove;
                    'L' :display(start);
                    'S' :save(text,fname,start);
                    'E' :editLine;
                    'Q' :done:=true;
            end;
        until done=true;
textcolor(15);textbackground(0);
end.
```

^Z

Appendix J.12

Dirlst

```
(*==========================================================================*)
Procedure Dirlst(Var FileName:Path);

=================================
  Produces Directory Listing of files
=================================

CONST

   Max = 14;
   NorF= 15;(* White - Bright *)
   NorB= 1; (* Blue          *)
   SelF= 0; (* Black         *)
   SelB= 7; (* White         *)
   StdF= 12;(* Red -   Bright *)
   LevF= 14;(* Yellow -Bright *)
VAR

   SelRec,TraceRec:            DPointer;
   I,Row,Col,WorkRow:              integer;

 (*==========================================================================*)
  Procedure chscr(F,B:integer);

  Begin
     TextColor(F);
     TextBackGround(B);
  End;

 (*==========================================================================*)
  Procedure PrtDir;

  VAR
     I:       integer;

  Begin

    Row:=1;  Col:=3;
    I:=1;
    While ((TraceRec <> Nil) and (Row < MAX)) Do Begin
           chscr(NorF,NorB);
           GotoXY(1,Row); ClrEOL;
           If WorkRow = Row Then Begin
```

```
            chscr(SelF,SelB);
            GotoXY(1,Row); ClrEOL;
        End
        Else Begin
            chscr(NorF,NorB);
            GotoXY(1,Row); ClrEOL;
        End;
        If SelRec = TraceRec Then Begin
           GotoXY(2,Row);
           Write('*');
        End;
        GotoXY(Col,Row);
        Write(TraceRec^.Name);
        GotoXY(Col + 16,Row);
        Write(TraceRec^.Size:11:0);
        GotoXY(Col + 36,Row);
        Write(TraceRec^.Month:2,'-');
        If TraceRec^.Day < 10 Then
          Write('0',TraceRec^.Day:1,'-')
        Else Write(TraceRec^.Day:2,'-');
        Write(TraceRec^.Year:2);
        GotoXY(Col + 49,Row);
        If TraceRec^.Hr > 12 Then
          Write(TraceRec^.Hr - 12:4,':')
        Else Write(TraceRec^.Hr:4,':');
        If TraceRec^.Minu < 10 Then
          Write('0',TraceRec^.Minu:1)
        Else Write(TraceRec^.Minu:2);
        If TraceRec^.Hr >=  12 Then
           Write('p')
        Else Write('a');
        Row:=Row + 1;
        Bottom:=TraceRec;
        TraceRec:=TraceRec^.Next;
        I:=I + 1;
  End;  (* While TraceRec <> Nil *)
  Chscr(NorF,NorB);
  If Row < MAX Then
   For I:=Row to MAX Do Begin
       GotoXY(1,I); ClrEOL;
   End;

End;  (* Procedure PrtDir *)
(*=================================================================*)
```

```
Procedure PrtRec(Point:DPointer;Row:integer);

Var
  Col: integer;

Begin
  chscr(NorF,NorB);
  Col:=3; GotoXY(1,Row); ClrEOL;
  If WorkRow = Row Then Begin
        chscr(SelF,SelB);
        GotoXY(1,Row); ClrEOL;
  End
  Else Begin
         chscr(NorF,NorB);
         GotoXY(1,Row); ClrEOL;
  End;
  If SelRec = Point Then Begin
     GotoXY(2,Row);
     Write('*');
  End;
  GotoXY(Col,Row);
  Write(Point^.Name);
  GotoXY(Col + 16,Row);
  Write(Point^.Size:11:0);
  GotoXY(Col + 36,Row);
  Write(Point^.Month:2,'-');
  If Point^.Day < 10 Then
     Write('0',Point^.Day:1,'-')
  Else Write(Point^.Day:2,'-');
  Write(Point^.Year:2);
  GotoXY(Col + 49,Row);
  If Point^.Hr > 12 Then
     Write(Point^.Hr - 12:4,':')
  Else Write(Point^.Hr:4,':');
  If Point^.Minu < 10 Then
     Write('0',Point^.Minu:1)
  Else Write(Point^.Minu:2);
  If Point^.Hr >=  12 Then
     Write('p')
  Else Write('a');
  chscr(NorF,NorB);

 End;  (* Procedure PrtRec *)
(*=====================================================================*)
```

```
Procedure GetFile;

LABEL  LEAVE;

VAR
  Opt:        char;
  CurRec:     DPointer;
  Check:      DPointer;
  I:          integer;
  Quit:       Boolean;
  TmpName:    String20;

Begin    (* Procedure GetFile *)

  Opt:=' ';
  CurRec:=Top;  WorkRow:=1;
  Quit:=False;
  While Quit = False Do Begin
     GotoXY(1,WorkRow);
     Read(Kbd,Opt);
     Case Ord(Opt) of
        27: Begin
              If KeyPressed Then Begin
              GotoXY(1,WorkRow);
              Read(Kbd,Opt);
              If NoFiles > 0 Then Case Ord(Opt) of
(* Help Message *) 60,59: Begin
                          TmpName:='';
                          TmpName:=CurRec^.Name;
                          Comments(CurRec^.Name);
                          Window(8,3,69,15);
                        End;
(* UP ARROW *)        72: If CurRec^.Prev <> Nil Then Begin
                          WorkRow:=WorkRow - 1;
                          If WorkRow >= 1 Then Begin
                             PrtRec(CurRec,WorkRow + 1);
                             PrtRec(CurRec^.Prev,WorkRow);
                          End
                          Else Begin
                             Top:=Top^.Prev;
                             TraceRec:=Top;
                             Bottom:=Bottom^.Prev;
                             PrtRec(CurRec,1);
                             GotoXY(1,1); InsLine;
                             WorkRow:=1;
                             PrtRec(CurRec^.Prev,WorkRow);
                          End;
```

```
                    CurRec:=CurRec^.Prev;
                  End
                  Else Write(^G);
(* PAGE UP *)     73: If Top^.Prev <> Nil Then Begin
                        I:=1;
                        While ((Top^.Prev <> Nil) and (I < 16)) Do Begin
                             Top:=Top^.Prev;
                             I:=I + 1;
                        End;
                        TraceRec:=Top;
                        WorkRow:=1;
                        PrtDir;
                        CurRec:=Top;
                  End
                  Else Write(^G);
(* DOWN ARROW *)  80: If CurRec^.Next <> Nil Then Begin
                     WorkRow:=WorkRow + 1;
                     If WorkRow < MAX  Then Begin
                        PrtRec(CurRec,WorkRow - 1);
                        PrtRec(CurRec^.Next,WorkRow);
                     End
                     Else Begin
                        Top:=Top^.Next;
                        Bottom:=Bottom^.Next;
                        TraceRec:=Top;
                        PrtRec(CurRec,WorkRow - 1);
                        Writeln;
                        WorkRow:=MAX -1;
                        PrtRec(CurRec^.Next,WorkRow);
                     End;
                     CurRec:=CurRec^.Next;
                  End
                  Else Write(^G);
(* PAGE DOWN *)   81: If Bottom^.Next <> Nil Then Begin
                        TraceRec:=Bottom;
                        Top:=Bottom;
                        WorkRow:=1;
                        PrtDir;
                        CurRec:=Top;
                  End
                  Else Write(^G);
             End;  (* Case Statement *)
            End (* Key Pressed *)
            Else Quit:=True;
           End;  (* 27 *)
```

```
          13: If NoFiles > 0 Then Begin
                Check:=Top;
                I:=1;
                While (( I < MAX) and (Check^.Next <> Nil)) Do Begin
                    If Check = SelRec Then Begin
                       SelRec:=CurRec;
                       PrtRec(Check,I);
                       Goto LEAVE;
                    End;
                    I:=I + 1;
                    Check:=Check^.Next;
                End;
LEAVE:Quit:=True;
                SelRec:=CurRec;
                PrtRec(SelRec,WorkRow);
              End;
        End;  (* Case Statement *)

     End; (* While Do *)

   End;     (* Procedure GetFile *)
  (*=================================================================*)

Begin
   if CrtMode=7 then   image:=monobuffer else image:=colorbuffer;
   chscr(NorF,NorB);
   window(2,2,79,24);
   ClrScr;

   GotoXY(8,1); Write('Directory of ',ChkDir);
   For I:=1 to 25 Do FileName[I]:=' ';
   GotoXY(35,1);
   Write(NoFiles:2,' File','(','s',') ');
   Diskspace;
      chscr(NorF,NorB);
   GotoXY(4,23);    Write(' Select:');
   GotoXY(23,23);   Write(' Up:');
   GotoXY(38,23);   Write(' Down:');
   GotoXY(55,23);   Write(' Exit:');
   GotoXY(66,23);   Write(' Help:');
   chscr(StdF,NorB);
   gotoxy(13,23);     Write('Enter ',Chr(17),Chr(196),Chr(217));
   gotoxy(28,23);     Write('PgUp or ',CHR(24));
   gotoxy(45,23);     Write('PgDn or ',CHR(25));
   gotoxy(62,23);     Write('Esc ');
   gotoxy(73,23);     Write('F1');
   If CrtMode = 7 Then
      NormVideo
```

```
    Else TextColor(NorF);
    Window(8,3,69,15);
    WorkRow:=1;
    TraceRec:=FirstRec;
    Top:=TraceRec;
    PrtDir;
    SelRec:=Nil;
    GetFile;
    ClrScr;
    If SelRec <> Nil Then Begin
       FileName:='';
       FileName:=SelRec^.Name;
    End;

    If CrtMode = 7 Then
       MonoBuffer:=Image
    Else ColorBuffer:=Image;

    Window(2,2,79,24);
    chscr(LevF,NorB);

End;
^Z

Procedure DisplayCom;

VAR
  J,ComNo,I,NoChar,Row: Integer;
  StartN,EndN: Integer;
  Found: Boolean;

Begin  (* Procedure DisplayCom *)

    TmpCom1:=''; TmpCom2:=''; TmpCom3:=''; TmpCom4:=''; TmpCom5:='';
    (* Break up Comments *)
    NoChar:=Length(TempCom);
    Row:=11;
    EndN:=0; ComNo:=1;
    While (EndN + 1 <=NoChar) Do Begin
       StartN:=EndN + 1; Found:=False;
       While ((StartN <=NoChar) and (Found = False)) Do Begin
          If TempCom[StartN] = ' ' Then StartN:=StartN + 1
          Else Found:=True;
       End;
       EndN:=StartN + 50;
       If EndN > NoChar Then EndN:=NoChar;
       (* Break up into words *)
```

```
      Found:=False;
      If ((EndN <> NoChar) and (TempCom[EndN + 1] <> ' ')) Then
         While ((EndN > StartN) and (Found = False)) Do Begin
             If TempCom[EndN] <> ' ' Then
                EndN:=EndN - 1
             Else Found:=True;
         End;

      If Default = False Then Begin
         GotoXY(15,Row);
         For I:=StartN to EndN Do
            Write(TempCom[I]);
      End;
      Case ComNo of
         1: TmpCom1:=Copy(TempCom,StartN,EndN - StartN + 1);
         2: TmpCom2:=Copy(TempCom,StartN,EndN - StartN + 1);
         3: TmpCom3:=Copy(TempCom,StartN,EndN - StartN + 1);
         4: TmpCom4:=Copy(TempCom,StartN,EndN - StartN + 1);
         5: TmpCom5:=Copy(TempCom,StartN,EndN - StartN + 1);
      End; (* Case Statement *)
      ComNo:=ComNo + 1;
      Row:=Row + 1;
   End;
   For I:=Length(TmpCom1) + 1 to 51 Do TmpCom1:=TmpCom1 + ' ';
   For I:=Length(TmpCom2) + 1 to 51 Do TmpCom2:=TmpCom2 + ' ';
   For I:=Length(TmpCom3) + 1 to 51 Do TmpCom3:=TmpCom3 + ' ';
   For I:=Length(TmpCom4) + 1 to 51 Do TmpCom4:=TmpCom4 + ' ';
   For I:=Length(TmpCom5) + 1 to 51 Do TmpCom5:=TmpCom5 + ' ';
End; (* Procedure DisplayCom *)
^Z
```

Appendix J.13

Diskspace

```
(*==========================================================================*)
Procedure Diskspace;

=================================================
      Get free disk space in bytes from DOS.
=================================================


Var
 Tracks,Sectors,Bytes: integer;
 Space: real;
 Reg: RegRec;

Begin

   If ChkDir <> ProgDrvDir Then
   ChDir(ChkDir);
   reg.ax:=$3600;
   reg.dx:=0;
   Intr($21,Reg);
   Tracks:=reg.bx;
   Sectors:=Reg.ax;
   Bytes:=Reg.cx;
   space:=((Sectors * Bytes * 1.0) * Tracks);
   If ChkDir <> ProgDrvDir Then
   ChDir(ProgDrvDir);

   write(space:10:0,' bytes free      ');

End;^Z
```

[BLANK PAGE]

Appendix J.14

run_me.1st

```
program run_me_1_st;
type s80 = string[79];
var
i,j,k,l,m,n,si,s,
e,ck,pr,nl,pg,tpg,
c1,c2,ch1,ch2,w,ww
                                        :integer;
lx                                  :array[1..803] of s80;
filein,fileout                       :text;
d                                    :char;
li                                  :s80;
procedure scrx;
begin
     clrscr;
     for j:=m to m+22 do
     begin
          ww:=length(lx[j]);if lx[j]=chr(12) then ww:=0;
          if (ww<80) and (ww>0) then writeln(lx[j]);
          if ww=80 then write(lx[j]);
          if ww=0 then writeln;
     end;
     write(lx[j+1]);
end;
function change(xx:integer):integer;
begin
     if xx<1 then xx:=1;
     if (xx>nl-23) then xx:=nl-23;
     change:=xx;
end;
procedure one;
begin
     writeln;
     write(lx[m+23]);
end;
begin
 randomize;c1:=random(2)+14;c2:=random(3)*2+1;
 textcolor(c1);textbackground(c2);clrscr;
 gotoxy(2,2);
 writeln('You can view manual on screen or print it.');
 Write(' Output Manual To Printer (Y/N): ');
 readln(d);
 pr:=0; if (d='Y') or (d='y') then pr:=1;
 assign(filein,'nail.trm');
```

```
 reset(filein);
 si:=1;i:=si-1;tpg:=0;w:=0;pg:=1; nl:=1;
 while not eof(filein) do
      begin
       i:=i+1;w:=w+1;
       readln(filein,li);
       if pr=1 then writeln(lst,li);
       k:=length(li);
       if (k>0) or (w<2)  then
          begin
               lx[nl]:=li;
               nl:=nl+1;
          end;
      if k>0 then w:=0;
      end;
 i:=i-1;            nl:=nl-1;
 close(filein);
 writeln;
 textbackground(7);textcolor(0);
 clrscr;
 writeln('      Use the 4 arrows, PgUp, and PgDn commands to view text and S to
stop');
 window(1,2,80,25);
 textcolor(c1);
 textbackground(c2);m:=1;ch1:=0;
 scrx;
 while (ch1<>83) and (ch1<>115) do
  begin
  ch1:=0;ch2:=0;
       if keypressed then
          begin
               read(kbd,d);
               ch1:=ord(d);
               if (ch1=27) and keypressed then
               begin
                    read(kbd,d);
                    ch2:=ord(d);
               end;
          if ch2=80 then
             begin
                  gotoxy(1,24);
                  m:=change(m+1);
                  one;
             end;
          if ch2=81 then
             begin
```

```
                m:=change(m+23);
                scrx;
            end;
        if ch2=73 then
            begin
                m:=change(m-23);
                scrx;
            end;
        if ch2=71 then
            begin
                m:=change(1);
                scrx;
            end;
        if ch2=79 then
            begin
                m:=change(nl);
                scrx;
            end;
        if ch2=72 then
            begin
                m:=change(m-1);
                gotoxy(1,1);
                insline;
                write(lx[m]);
            end;
        if ch2=75 then
            begin
                m:=change(m-1);
                while (lx[m]<>chr(12)) and (m>1) do
                m:=change(m-1);
                scrx;
            end;
        if ch2=77 then
            begin
                m:=change(m+1);
                while (lx[m]<>chr(12)) and (m<nl-23) do
                m:=change(m+1);
                scrx;
            end;

        end;

  end;
end.
^Z
```

[BLANK PAGE]

Appendix J.15

Exists

```
Function Exists(Name:Path):Boolean;
(*-----------------------------------*)
(* Determine if File (FilName) Exists *)
(*-----------------------------------*)

Var
  Fil: File;

Begin  (* Function Exists *)

  Assign(Fil,Name);
  {$I-}
  Reset(Fil);
  Close(Fil);
  {$I+}
  Exists:=(IOresult = 0);

End;  (* Function Exists *)
^Z
```

[BLANK PAGE]

Appendix J.16

FileName

```
Procedure FileName(Var Name:Path;Title:String10);
 (*-------------------------------------*)
 (* Read File Name from Terminal Screen  *)
 (*-------------------------------------*)

 Label TRYAGAIN;

 Var
   Col,I,J: integer;
   StartSt,EndSt: integer;
   Ch: char;

 Begin (* Procedure FileName *)

TRYAGAIN:
   TextBackGround(Blue);
   TextColor(Yellow);
   ClrScr;
   GotoXY(12,8); Write('  Enter ',Title,' File Name: ______________________________________');
   GotoXY(14,23); Write('Directory: Enter(',Chr(17),chr(196),chr(217),')');
   GotoXY(54,23); Write('Abort: Crtl Z');
   If ReqFile = False Then Begin
     GotoXY(14,12);
     Write('Press Esc to Exit');
   End;
   Col:=42;
   For I:=1 to 25 Do Begin
       If Name[I] <> ' ' Then Begin
          GotoXY(Col,8);
          Write(Name[I]);
       End;
       Col:=Col + 1;
   End;
   Col:=42; I:=1;
   Ch:=' ';
   While ((I <= 26) and (ord(ch) <> 13)) Do Begin
     GotoXY(Col,8); Read(kbd,Ch);
     If Ord(Ch) = 13 Then Begin
         If Name = '                        ' Then Begin
            DirLst(Name);
            If (Name <> '                        ') Then Begin
               Col:=42;
               For I:=1 to 25 Do Begin
```

```
                    If Name[I] <> ' ' Then Begin
                        GotoXY(Col,8);
                        Write(Name[I]);
                    End;
                    Col:=Col + 1;
                 End;
                 Col:=42;
                 Ch:=' ';
                 I:=1;
              End
              Else Ch:=' ';
           End;
        End
        Else If CH = ^Z
 Then Begin
           ClrScr;
           GotoXY(27,10);
           Release(HeapTop);
           Write(^G,'*** PROGRAM ABORTED ***');
           Delay(3000);
           Halt;
        End
        Else If ((CH = ':') and (I = 2)) Then Begin
             GotoXY(Col,8); Write(Ch);
             Name[I]:=ch;
             Col:=Col + 1; I:=I + 1;
        End
        Else Case Ord(Ch) of
                    27: Begin
                          If KeyPressed Then Begin
                           Read(Kbd,ch);
                           Case Ord(Ch) of
                              59: Help('Enter ' + Title + ' file name.',
                              'Press the Enter key (' + Chr(17) + chr(196) + chr(217) + ') to get a
directory listing.');
                              75: If I > 1 Then Begin
                                     I:=I - 1;
                                     Col:=Col - 1;
                                  End;
                              77: If I < 25 Then Begin
                                     I:=I + 1;
                                     Col:=Col + 1;
                                  End;
                              83: Begin
                                     For J:=I to 24 Do Begin
                                        Name[J]:=Name[J + 1];
                                        If Name[J] <> ' ' Then Write(Name[J])
```

```
                                    Else Write('_');
                                  End;
                                  Name[25]:=' ';
                                  Write('_');
                              End;
                            End; (* Case Statement  27 *)
                          End (* KeyPressed *)
                          Else If ReqFile = False Then I:=100;
                        End;
                    8: Begin
                          If Col > 42 Then Col:=Col - 1;
                          If I > 1 Then I:=I - 1;
                          GotoXY(Col,8); Write('_');
                          If I < 25 Then
                             For J:=I to 24 Do Begin
                                 Name[J]:=Name[J + 1];
                                 If Name[J] <> ' ' Then Write(Name[J])
                                 Else Write('_');
                             End;
                          Name[25]:=' ';
                          Write('_');
                        End;
46,48..57,65..90,92,97..122: If I <= 25 Then Begin
                          If (I = 1) Then Begin
                             GotoXY(2,6); ClrEOL;
                          End;
                          GotoXY(Col,8); Write(Ch);
                          Name[I]:=ch;
                          Col:=Col + 1; I:=I + 1;
                        End;
      End; (* Of Case *)
    End; (* Of While *)

    If ((Name = '                    ') and (ReqFile = True)) Then Begin
        ClrScr;
        GotoXY(15,12); Write(^G,'File Name Required !!');
        Delay(3000);
        Goto TRYAGAIN;
    End;

    (* REMOVE EMBEDDED BLANKS *)
    StartSt:=1;
    EndSt:=25;
    While ((Name[EndSt] = ' ') and (EndSt > 0)) Do EndSt:=EndSt - 1;
    If EndSt > 0 Then Repeat
```

```
      If Name[EndSt] = ' ' Then Begin
         For J:=StartSt to EndSt - 1 Do Name[J]:=Name[J + 1];
         Name[EndSt]:=' ';
         EndSt:=EndSt - 1;
      End
      Else StartSt:=StartSt + 1;
   Until (StartSt >= EndSt);

   End; (* Procedure FileName *)
^Z
```

Appendix J.17

FindCount

```
Program FindCount(Input,Output);
{$C-}
```

STEP 10 - Find & Count Data. This program is used to Evaluate the number of occurences of data in a given data set.

Deborah A. Holder

```
TYPE
   HoldName = array [1..25] of char;
   DateTimeType = String[8];
   RegRec = record
              ax,bx,cx,dx,bp,si,di,ds,es,flags: integer;
            end;
   String20 = string[20];
   String10 = string[10];
   Char12arr = array [1..12] of char;
   String255 = string[255];
   String50  = string[50];
   String6 = string[6];
   DPointer = ^DirRec;
   DirRec = record
              Size: real;
              Hr,Minu: integer;
              Year,Month,Day: integer;
              Next,Prev: DPointer;
              Name: String20;
            end;
   FPointer = ^FieldRec;
   SPointer = ^FieldRec;
   FieldRec = record
               Fields: String255;
               Occurences: real;
               Next: FPointer;
             end;
   SortRec = record
               Fields: String255;
               Occurences: real;
             end;
```

```
   ImageType = array [1..4096] of char;
   Path = string[64];

VAR
   HeapTop: ^Integer;
   NoFiles: integer;
   Top,Bottom: DPointer;
   FirstRec,LastRec,NewRec: DPointer;
   FirstFieldRec,LastFieldRec: array [1..5] of FPointer;
   FirstSortRec: SPointer;
   NewSortRec: SPointer;
   NewFieldRec: FPointer;
   IFileName: Path;
   GFN,InFile: Text;
   ReqFile,ValFile: Boolean;
   J,K,I: integer;
   BegCol,EndCol: array [1..5] of integer;
   CombineWith:  array [1..5] of integer;
   MaxRec: array [1..5] of integer;
   NumFields: integer;
   Line: String255;
   LineArr: array [1..5] of String255;
   SortNo: integer;
   SortAns: Char;
   HelpMsg: integer;
   Image: ImageType;
   CrtMode : Byte absolute $0040:$0049;
   MonoBuffer: ImageType absolute $B000:$0000;
   ColorBuffer: ImageType absolute $B800:$0000;
   ChkDir,ProgDrvDir,DrvDir: Path;

{$IBoxWin.pas}
{$IHelp.pas}
{$IAddDir.pas}
{$IComments.pas}
{$IExists.pas}
{$IDiskSpace.pas}
{$IGetDir.pas}
{$IDirLst.pas}
{$IFileName.pas}
{$IDate.pas}
{$ITime.pas}

(*=====================================================================*)
```

```
Procedure GetCol;   (* Find Beginning and Ending Columns *)

TYPE
  String3 = string[3];
VAR
  I: integer;
  BegCh,EndCh,ComCh: String3;
  Row,Col: integer;
  Limit,Code: integer;
  ChkVal: integer;


 (*==============================================================*)
   Procedure ReadCol(Var ChStr: String3);

   VAR
     J: integer;
     Ch: char;


   Begin  (* Procedure ReadCol *)
      ChStr:='';
      CH:=' ';
      J:=1;
      While ((Ord(Ch) <> 13) and (J <=Limit + 1)) Do Begin
           GotoXY(Col,Row);
           Read(kbd,ch);
           Case Ord(Ch) of
                  27: Begin
                         If KeyPressed Then Begin
                            Read(Kbd,ch);
                            If Ord(Ch) = 59 Then
                             Case HelpMsg of
                               1: Help('Enter the Beginning Column of the field.','');
                               2: Help('Enter the Ending Column of the field.','');
                               3: Help('Enter the Field number to be combined.','');
                             End;  (* Case HelpMsg *)
                         End;
                      End;
             48..57: If J <=Limit Then Begin
                         GotoXY(Col,Row);
                         Write(Ch);
                         Insert(Ch,ChStr,J);
                         Col:=Col + 1;
                         J:=J + 1;
                      End;
```

```
                8: If J > 1 Then Begin
                     Col:=Col - 1;
                     J:=J - 1;
                     Delete(ChStr,J,1);
                     GotoXY(Col,Row);
                     Write('_');
                   End;

          End;
     End;

   End;   (* Procedure ReadCol *)

 (*==========================================================*)

Begin (* Procedure GetCol *)
  I:=1;Row:=4;
  BegCh:=''; EndCh:='';
  Limit:=3;
  GotoXY(26,1);
  Write('BEGINNING AND ENDING COLUMNS');
  GotoXY(10,22);
  Write('Press  RETURN  after entering all starting and ending columns');
  Repeat
    GotoXY(20,Row);
    Write('Field number ',I:1,' starts in column: _____');
    Col:=53; HelpMsg:=1;
    ReadCol(BegCh);
    Val(BegCh,BegCol[I],Code);
    If BegCol[I] <> 0 Then Begin
       NumFields:=I;
       Row:=Row + 1;
       GotoXY(20,Row);
       Write('Field number ',I:1,' ends in column:   _____');
       Col:=53; HelpMsg:=2;
       ReadCol(EndCh);
       Val(EndCh,EndCol[I],Code);
       If ((EndCol[I] = 0) or (EndCol[I] < BegCol[I]))
           Then EndCol[I]:=BegCol[I];
    End;
    Row:=Row + 2;
    I:=I + 1;
  Until ((I > 5) or (BegCol[I - 1] = 0));

  ClrScr; Row:=8; Limit:=1; I:=1;
  GotoXY(31,2);
  Write('COMBINED FIELDS');
```

```
    GotoXY(10,22);
    Write('Press  RETURN  after entering all combined fields');
    Repeat
       GotoXY(15,Row);
       Write('Combine Field Number ',I:1,' with Field Number : __');
       Col:=58; HelpMsg:=3;
       ReadCol(ComCh);
       ChkVal:=0;
       Val(ComCh,ChkVal,Code);
       If ChkVal <= NumFields Then Begin
          CombineWith[I]:=ChkVal;
          Row:=Row + 2;
          I:=I + 1;
       End
       Else Begin
          GotoXY(1,21); ClrEOL;
          GotoXY(15,21); Write(^G,'Invalid Field !!  Field was not defined');
       End;
    Until (I > NumFields);

  End;  (* Procedure GetCol *)
(*=========================================================================*)
  Procedure BreakAndCombine;

  VAR
    I: integer;

  Begin   (* Procedure BreakUpLine *)

    (* Break up String *)
    For I:=1 to NumFields Do
        LineArr[I]:=Copy(Line,BegCol[I],EndCol[I] - BegCol[I] + 1);

    (* Combine proper Strings *)
    For I:=1 to NumFields Do Begin
        If CombineWith[I] <> 0 Then
           LineArr[I]:=LineArr[I] + LineArr[CombineWith[I]];
    End;

  End;    (* Procedure BreakUpLine *)
(*=========================================================================*)

  Procedure FindOccurences;

    VAR
      I: integer;
      TrackField: FPointer;
      Found: Boolean;
```

```
  Begin (* Procedure FindOccurences *)

     For I:=1 to NumFields Do Begin
        Found:=False;
        TrackField:=FirstFieldRec[I];
        While ((TrackField <> Nil) and (Found = False)) Do Begin
              If LineArr[I] = TrackField^.Fields Then Begin
                 Found:=True;
                 TrackField^.Occurences:=TrackField^.Occurences + 1;
              End;
              TrackField:=TrackField^.Next;
        End; (* While Loop *)
        If Found = False Then Begin
           New(NewFieldRec);
           MaxRec[I]:=MaxRec[I] + 1;
           NewFieldRec^.Fields:='';
           NewFieldRec^.Occurences:=0;
           NewFieldRec^.Fields:=LineArr[I];
           NewFieldRec^.Occurences:=1;
           If FirstFieldRec[I] = Nil Then
              FirstFieldRec[I]:=NewFieldRec
           Else LastFieldRec[I]^.Next:=NewFieldRec;
           LastFieldRec[I]:=NewFieldRec;
           LastFieldRec[I]^.Next:=Nil;
        End;
     End; (* For Loop *)

  End;  (* Procedure FindOccurences *)
(*=========================================================================*)
  Procedure PrintOccurences;

    VAR
      J,I: integer;
      Max: integer;
      TrackRec: array [1..5] of FPointer;
      PrtTime,PrtDate: DateTimeType;
      PrtLen: integer;

  Begin (* Procedure PrintOccurences *)

      ClrScr;
      GotoXY(27,10);
      Write(^G,'PRINTING RESULTS');
      Max:=MaxRec[1];
      For I:=2 to NumFields Do
          If MaxRec[I] > Max Then Max:=MaxRec[I];
```

```
    For I:=1 to NumFields Do TrackRec[I]:=FirstFieldRec[I];
    Writeln(Lst,^L);
    PrtDate:=Date;
    PrtTime:=Time;
    Writeln(Lst,'Date : ',PrtDate);
    Write(Lst,'Time : ',PrtTime);
    Writeln(Lst,'File : ',IFileName);
    For I:=1 to NumFields Do Begin
        Write(Lst,' field',I:2,' col. (',BegCol[I]:3,' - ',EndCol[I]:3,')');
        If CombineWith[I] <> 0 Then Write(Lst,' & combined with field',CombineWith[I]:3);
        Writeln(Lst);
    End;
    Writeln(lst);
    Write(Lst,' ':7);
    For I:=1 to NumFields Do Begin
        PrtLen:=EndCol[I] - BegCol[I] + 1;
        If CombineWith[I] <> 0 Then
        PrtLen:=PrtLen + EndCol[CombineWith[I]] - BegCol[CombineWith[I]] + 1;
        Write(Lst,'Field #',I:PrtLen,'   ');
    End;
    Writeln(Lst);
    Write(Lst,' Numb. ');
    For I:=1 to NumFields Do Begin
        Write(Lst,'[count] ');
        PrtLen:=EndCol[I] - BegCol[I] + 1;
        If CombineWith[I] <> 0 Then
        PrtLen:=PrtLen + EndCol[CombineWith[I]] - BegCol[CombineWith[I]] + 1;
        For J:=1 to PrtLen Do
            Write(Lst,'*');
        Write(Lst,'  ');
    End;
    Writeln(Lst);
    For J:=1 to Max Do Begin
       Write(lst,J:5,'.');
       For I:=1 to NumFields Do Begin
           PrtLen:=EndCol[I] - BegCol[I] + 1;
           If CombineWith[I] <> 0 Then
           PrtLen:=PrtLen + EndCol[CombineWith[I]] - BegCol[CombineWith[I]] + 1;
           If TrackRec[I] <> Nil Then Begin
              Write(Lst,'  [',TrackRec[I]^.Occurences:4:0,'] ');
              Write(lst,TrackRec[I]^.Fields:PrtLen,' ')
           End
           Else Write(lst,' ':10,' ':PrtLen);
       End;
       Writeln(Lst);
       For I:=1 to NumFields Do
           If TrackRec[I] <> Nil Then TrackRec[I]:=TrackRec[I]^.Next;
   End;
End;  (* Procedure PrintOccurences *)
```

```
(*==========================================================================*)
  Procedure SortArr;

  VAR
    TrackRec: FPointer;
    Cur,Prev,CPrev,Comp: FPointer;
    HCur:FPointer;

  Begin (* Procedure SortArr *)

     Cur:=FirstFieldRec[SortNo];
     HCur:=Cur;
     Prev:=Nil;
     CPrev:=Cur;
     Comp:=Cur^.Next;
     Repeat
          If Comp <> Nil Then Repeat
            If Cur^.Fields < Comp^.Fields Then Begin
               CPrev:=Comp;
               Comp:=Comp^.Next;
            End
            Else Begin
               CPrev^.Next:=Comp^.Next;
               Comp^.Next:=Cur;
               If Prev <> Nil Then Prev^.Next:=Comp;
               If Cur = HCur Then HCur:=Comp;
               Cur:=Comp;
               CPrev:=Cur;
               Comp:=Cur^.Next;
            End;
          Until (Comp = Nil);
          Prev:=Cur;
          Cur:=Cur^.Next;
          CPrev:=Cur;
          Comp:=Cur^.Next;
     Until Cur = Nil;

     FirstFieldRec[SortNo]:=HCur;
  End;  (* Procedure SortArr *)
(*==========================================================================*)
  Procedure SortRecords;

    VAR
      I: integer;
```

```
  Begin
     ClrScr;
     GotoXY(29,10);
     Write(^G,'SORTING FIELDS');
     For I:=1 To NumFields Do Begin
          SortNo:=I;
          SortArr;
     End;

  End;
(*==========================================================================*)

Begin  (* Program FindCount *)

    GraphBackGround(Blue);
    TextBackGround(Blue);
    ClrScr;
    BoxWin(1,1,80,24,'  STEP 10 ');
    GotoXY(26,10);
    Write('>>> Find & Count Data <<<');
    GotoXY(1,1);
    GetDir(0,ProgDrvDir);
    Assign(GFN,'GFN_Nail');
    Reset(GFN);
    ChkDir:='';
    Readln(GFN,DrvDir);
    Close(GFN);
    ChkDir:=DrvDir;
    If Length(ChkDir) = 2 Then ChkDir:=ChkDir + '\';
    Delay(1000);
    Mark(HeapTop);
    GetDirectory('????????.D??');
    ValFile:=False;
    ReqFile:=True;
    While ValFile = False Do Begin
        IFileName:='                        ';
        FileName(IFileName,'  Input   ');
        If ProgDrvDir <> DrvDir Then AddDir(IFileName);
        ValFile:=Exists(IFileName);
        If ValFile = False Then Begin
           GotoXY(1,12); ClrEOL;
           GotoXY(15,12); Write(^G,'File Does Not Exist');
           Delay(3000);
        End;
    End;
    Release(HeapTop);
    Assign(InFile,IFileName);
```

```
    ReSet(InFile);
    For I:=1 to 5 Do Begin
        BegCol[I]:=0; EndCol[I]:=0;
        CombineWith[I]:=0;
    End;
    NumFields:=0;
    I:=1;
    ClrScr;
    GetCol;
    ClrScr;
    Mark(HeapTop);
    For I:=1 to 5 Do Begin
        FirstFieldRec[I]:=Nil;
        MaxRec[I]:=0;
    End;
    While (NOT EOF(InFile)) Do Begin
       Readln(InFile,Line);
       BreakAndCombine;
       FindOccurences;
    End;
    ClrScr;
    GotoXY(18,10);
    Write('Do wish to Sort fields? [Y,N]: _');
    K:=WhereX - 1;  J:=WhereY;
    Repeat
       SortAns:=' ';
       GotoXY(K,J);
       Read(Kbd,SortAns);
       If Ord(SortAns) = 27 Then Begin
          If KeyPressed Then Read(Kbd,SortAns);
          If Ord(SortAns) = 59 Then
             Help('Enter  Y  if you wish the fields selected to be sorted.','');
       End
       Else Begin
          SortAns:=UpCase(SortAns);
          GotoXY(K,J);
          Write(SortAns);
          If ((SortAns <> 'Y') and (SortAns <> 'N')) Then Begin
             GotoXY(10,15); Write(^G,'Invalid Response');
          End;
       End;
    Until ((SortAns = 'Y') or (SortAns = 'N'));
    If SortAns = 'Y' Then SortRecords;
    PrintOccurences;
    Close(InFile);
    Release(HeapTop);
    Window(1,1,80,25);
    ClrScr;
End.
^Z
```

Appendix J.18

FixR710

```
Program FixR710(Input,Output);
{$C-}
```

STEP 13 - Fix R710. This program is used to reduce the R710 report to a simplified data set.

Deborah A. Holder

```
TYPE
   HoldName  = array [1..25] of char;
   RegRec = record
               ax,bx,cx,dx,bp,si,di,ds,es,flags: integer;
            end;
   String255 = string[255];
   String20  = string[20];
   String50  = string[50];
   Char12arr = array [1..12] of char;
   String10  = string[10];
   String3   = string[3];
   String6   = string[6];
   DPointer = ^DirRec;
   DirRec = record
              Size: real;
              Hr,Minu: integer;
              Year,Month,Day: integer;
              Next,Prev: DPointer;
              Name: String20;
            end;
  ImageType = array [1..4096] of char;
  Path = string[64];

VAR
   HeapTop: ^Integer;
   K,I,NoFiles: integer;
   Check,Lc,Code: integer;
   Top,Bottom: DPointer;
   FirstRec,LastRec,NewRec: DPointer;
   TFile,OFileName,IFileName: Path;
   GFN,InFile,OutFile:Text;
   Line: String255;
```

```
   Default,Found,ReqFile,ValFile: Boolean;
   Ans: char;
   Month: array [1..12] of string6;
   ChkMonth,Year: string6;
   BY,Y: real;
   Image: ImageType;
   Field,MaxField: Integer;
   UppCase,Quit: Boolean;
   CrtMode: Byte absolute $0040:$0049;
   MonoBuffer: ImageType absolute $B000:$0000;
   ColorBuffer: ImageType absolute $B800:$0000;
   ChkFile,ChkDir,ProgDrvDir,DrvDir: Path;
   TempCom: String255;
   TmpCom1,TmpCom2,TmpCom3,TmpCom4,TmpCom5: Path;
   HlpMsg1,HlpMsg2: String255;


{$IBoxwin.pas}
{$IHelp.pas}
{$IAddDir.pas}
{$IComments.pas}
{$IExists.pas}
{$IDiskSpace.pas}
{$IGetDir.pas}
{$IDirLst.pas}
{$IFileName.pas}
{$IAddFileExt.pas}
{$IReadField.pas}
{$IReadCom.pas}
{$IDisCom.pas}
{$IUpdateCom.pas}
{$ICreatSqr.pas}

(*=====================================================================*)
   Procedure InitMonth;

   Begin  (* Procedure InitMonth *)

     MONTH[ 1]:='JAN';  MONTH[ 2]:='FEB';  MONTH[ 3]:='MAR';  MONTH[ 4]:='APR';
     MONTH[ 5]:='MAY';  MONTH[ 6]:='JUN';  MONTH[ 7]:='JUL';  MONTH[ 8]:='AUG';
     MONTH[ 9]:='SEP';  MONTH[10]:='OCT';  MONTH[11]:='NOV';  MONTH[12]:='DEC';

   End;  (* Procedure InitMonth *)
(*=====================================================================*)
```

```
Begin
    (*  Open Input File *)

    GraphBackGround(Blue);
    TextBackGround(Blue);
    TextColor(Yellow);
    ClrScr; Default:=False;
    BoxWin(1,1,80,24,'  STEP 13 ');
    GotoXY(31,10);
    Write('>>> Fix R710 <<<');
    GotoXY(1,1);
    GetDir(0,ProgDrvDir);
    Assign(GFN,'GFN_Nail');
    Reset(GFN);
    ChkDir:='';
    Readln(GFN,DrvDir);
    Close(GFN);
    ChkDir:=DrvDir;
    If Length(ChkDir) = 2 Then ChkDir:=ChkDir + '\';
    Delay(1000);
    Mark(HeapTop);
    GetDirectory('????????.dat');
    ClrScr;
    GotoXY(10,2);
    Write('This program will CHANGE the input file of R710 data into');
    GotoXY(10,3);
    Write('                    a compressed file.');
    Delay(3000);
    ValFile:=False;
    While ValFile = False Do Begin
       IFileName:='                          ';
       ReqFile:=True;
       FileName(IFileName,'  Input   ');
       AddFileExt(IFileName,'DAT');
       If ProgDrvDir <> DrvDir Then AddDir(IFileName);
       ValFile:=Exists(IFileName);
       If ValFile = False Then Begin
          GotoXY(1,12); ClrEOL;
          GotoXY(15,12); Write(^G,'File Does Not Exist !!');
          Delay(2000);
       End;
    End;
    Release(HeapTop);
    Assign(InFile,IFileName);
    Reset(InFile);
    TFile:='Temp.zzz';
    If ProgDrvDir <> DrvDir Then AddDir(TFile);
```

```
Assign(OutFile,TFile);
Rewrite(OutFile);
InitMonth;

LC:=0;
ClrScr; Found:=False;
While ((Found = False) and (Not EOF(InFile))) Do Begin
    Readln(InFile,Line);
    If Pos('REPORT PERIOD:',Line) <> 0 Then Begin
       Found:=True;
       K:=Pos(':',Line);
       ChkMonth:=Copy(Line,K+3,3);
       Year:=Copy(Line,K+7,2);

       For I:=1 to 12 Do
           If ChkMonth = Month[I] Then K:=I;

       For I:=1 to Length(Year)
           Do If Year[I] = ' ' Then Delete(Year,I,1);

       Val(Year,Y,Code);
       BY:=Y;
       Y:=30.42 * K;
       Str(Y:4:0,Year);
       Writeln(OutFile,Year,BY:4:0);
    End;
End;
While (Not EOF(InFile)) Do Begin
    Readln(InFile,Line);
    Lc:=Lc + 1;
    K:=1;  Found:=False;
    While ((K <=12) and (Found = False)) Do Begin
        Check:=Pos(Month[K],Line);
        If ((Check <> 0) and (Check < 5)) Then Found:=True
        Else K:=K + 1;
    End;
    If Found = True Then Begin
        If Check <> 1 Then
           Delete(Line,Check - 1,1);
        Year:=Copy(Line,5,4);
        For I:=1 to Length(Year)
            Do If Year[I] = ' ' Then Delete(Year,I,1);

        Val(Year,Y,Code);
        Y:=(Y - BY) * 365.0 + 30.42 * K;
        Str(Y:4:0,Year);
        Delete(Line,1,20);
```

```
            Insert(Year + ' ',Line,1);
            Delete(Line,11,10);
            Delete(Line,17, 6);
            Delete(Line,23, 8);
            Delete(Line,29, 9);
            Delete(Line,35, 6);
            Delete(Line,41, 9);
            Delete(Line,47, 3);
            Writeln(OutFile,Line);
            Write(lc,' ',Month[k],' ');
        End;
        If Pos('ORG :',Line) > 0 Then Begin
            Writeln;
            I:=1;
            While Line[I] = ' ' Do Delete(Line,I,1);
            Writeln(OutFile,Line);
            Writeln(Line);
        End;
    End;
    Close(InFile);
    Close(OutFile);
    CreateSqr;

End.^Z
```

[BLANK PAGE]

Appendix J.19

FormatRA

```
Program FormatRA(input,output);
{$C-}
```

STEP 19 - Format data for Regression Analysis. This step changes the output file created by Step 2 to a compressed format required for input to Step 4.

Deborah A. Holder

```
TYPE

    Str160 = string[160];
    HoldName = array [1..25] of char;
    RegRec = record
                ax,bx,cx,dx,bp,si,di,ds,es,flags: integer;
             end;
    Char12arr = array [1..12] of char;
    String255 = string[255];
    String50 = String[50];
    String20 = string[20];
    String10 = string[10];
    String3  = string[3];
    String6 = string[6];
    DPointer = ^DirRec;
    DirRec = record
               Size: real;
               Hr,Minu: integer;
               Year,Month,Day: integer;
               Next,Prev: DPointer;
               Name: String20;
             end;
    ImageType = array [1..4096] of char;
    Path = string[64];

VAR

    GFN,InFile,OutFile: Text;
    ChkFile,IFileName,OFileName: Path;
    HeapTop: ^Integer;
    Code,I,J,K,NoFiles: integer;
```

```
    Top,Bottom: DPointer;
    FirstRec,LastRec,NewRec: DPointer;
    UppCase,Default,Quit,ReqFile,ValFile: Boolean;
    ZeroAns,Ans: Char;
    Field,MaxField: Integer;
    F0Str,F1Str:String20;
    F0,F1: real;
    V0Str,V1Str,V2Str,V3Str,V4Str: String20;
    Line: Str160;
    Image: ImageType;
    CrtMode: Byte absolute $0040:$0049;
    MonoBuffer: ImageType absolute $B000:$0000;
    ColorBuffer: ImageType absolute $B800:$0000;
    ChkDir,ProgDrvDir,DrvDir: Path;
    TempCom: String255;
    TmpCom1,TmpCom2,TmpCom3,TmpCom4,TmpCom5: Path;
    HlpMsg1,HlpMsg2: String255;


{$IBoxWin.pas}
{$IHelp.pas}
{$IAddDir.pas}
{$IComments.pas}
{$IExists.pas}
{$IDiskSpace.pas}
{$IGetDir.pas}
{$IDirLst.pas}
{$IFileName.pas}
{$IAddFileExt.pas}
{$IReadField.pas}
{$IReadCom.pas}
{$IDisCom.pas}
{$IUpdateCom.pas}

Begin

    GraphBackGround(Blue);
    TextBackGround(Blue);
    TextColor(Yellow);
    Default:=False;
    ClrScr;
    BoxWin(1,1,80,24,'  STEP 19 ');
    GotoXY(17,10);
    Write('>>> Format Data for Regression Analysis <<<');
    GotoXY(1,1);
    GetDir(0,ProgDrvDir);
    Assign(GFN,'GFN_Nail');
```

```
Reset(GFN);
ChkFile:=''; ChkDir:=''; DrvDir:='';
Readln(GFN,DrvDir);
Readln(GFN,ChkFile);
Close(GFN);
ChkDir:=DrvDir;
If Length(ChkDir) = 2 Then ChkDir:=ChkDir + '\';
Delay(1000);
If ((ChkFile <> '**') and (Pos('DTC',ChkFile) >0)) Then Begin
   Default:=True;
   IFilename:=''; IFileName:=ChkFile;
   If DrvDir <> ProgDrvDir Then AddDir(IFileName);
End
Else Begin
   Mark(HeapTop);
   GetDirectory('????????.DTC');
   ValFile:=False;
   While ValFile = False Do Begin
      IFileName:='                    ';
      ReqFile:=True;
      FileName(IFileName,' Input  ');
      AddFileExt(IFileName,'DTC');
      If ProgDrvDir <> DrvDir Then AddDir(IFileName);
      ValFile:=Exists(IFileName);
      If ValFile = False Then Begin
         GotoXY(1,12); ClrEOL;
         GotoXY(15,12); Write(^G,'File Does Not Exists');
         Delay(3000);
      End;
   End;
   Release(HeapTop);
End;
If ((ChkFile <> '**') and (Pos('DTC',ChkFile) >0)) Then Begin
   OFilename:=''; OFileName:=ChkFile;
   AddFileExt(OFileName,'DTD');
   ChkFile:=OFileName;
   Assign(GFN,'GFN_Nail');
   ReWrite(GFN);
   Writeln(GFN,DrvDir);
   Writeln(GFN,ChkFile);
   Close(GFN);
   If DrvDir <> ProgDrvDir Then AddDir(OFileName);
End
Else Begin
   Mark(HeapTop);
   GetDirectory('????????.DTD');
```

```
   (* Get Output File and open File *)
   ValFile:=True;
   Repeat
      OFileName:='                    ';
      ReqFile:=True;
      FileName(OfileName,' Output ');
      AddFileExt(OFileName,'DTD');
      If ProgDrvDir <> DrvDir Then AddDir(OFileName);
      ValFile:=Exists(OFileName);
      If ValFile = True Then Begin
         GotoXY(1,12); ClrEOL;
         GotoXY(1,13); ClrEOL;
         GotoXY(17,12); Write(^G,'File Already Exist !!');
         GotoXY(17,13); Write('Do you wish to overwrite ?: _');
         GotoXY(WhereX -1,WhereY);
         Read(kbd,Ans);
         Ans:=UpCase(Ans);
         GotoXY(WhereX,WhereY);
         Write(Ans);
         If Ans = 'Y' Then ValFile:=False;
      End;
   Until (ValFile = False);
   Release(HeapTop);
End;

Assign(InFile,IFileName);
ReSet(InFile);
Assign(OutFile,OFileName);
ReWrite(OutFile);

If Default = False Then Begin
   ClrScr;
   GotoXY(17,10);
   Write('Do you wish to exclude zeros [Y/N]: _');
   J:=WhereX - 1;  K:=WhereY;
   Repeat
      ZeroAns:=' ';
      GotoXY(J,K);
      Read(kbd,ZeroAns);
      If Ord(ZeroAns) = 27 Then Begin
         Read(Kbd,ZeroAns);
         If Ord(ZeroAns) = 59 Then Help('Enter  Y  if you wish to exclude zeros from the file.',
              'Enter  N  if you wish to include zeros.');
      End
      Else Begin
         ZeroAns:=Upcase(ZeroAns);
         GotoXY(J,K);
```

```
          Write(ZeroAns);
          If ((ZeroAns <> 'Y') and (ZeroAns <> 'N')) Then Begin
             GotoXY(10,15); Write(^G,'Invalid Response');
          End;
       End;
    Until ((ZeroAns = 'N') or (ZeroAns = 'Y'));
End
Else ZeroAns:='Y';
ClrScr;
GotoXY(17,10);
Write('FORMATTING FILE FOR REGRESSION ANALYSIS');
Code:=5;
Writeln(OutFile,Code);
While (NOT EOF(InFile)) Do Begin
   Line:='';
   Readln(InFile,Line);
   F0Str:=Copy(Line,123,3);
   F1Str:=Copy(Line,126,6);
   I:=1;
   Repeat
     If F0Str[I] = ' ' Then
        Delete(F0Str,I,1)
     Else I:=I + 1;
   Until (I >= Length(F0Str));
   I:=1;
   Repeat
     If F1Str[I] = ' ' Then
        Delete(F1Str,I,1)
     Else I:=I + 1;
   Until (I >= Length(F1Str));
   Val(F0Str,F0,Code);
   Val(F1Str,F1,Code);
   If ((ZeroAns <> 'Y') or ((ZeroAns = 'Y') and (F1 >=1))) Then Begin
      V0Str:=Copy(Line,56,7);
      V1Str:=Copy(Line,132,6);
      V2Str:=Copy(Line,138,6);
      V3Str:=Copy(Line,144,6);
      V4Str:=Copy(Line,150,6);
      If F0 <> 0 Then Begin
         F1:=F1 + F0 * 0.01;
         F1Str:='';
         Str(F1:12:10,F1Str);
      End
      Else Begin
         J:=Length(F1Str);
         If J < 6 Then
            For I:=1 to 6 - J Do
```

```
            Insert(' ',F1Str,1);
        End;
        Writeln(OutFile,F1Str);
        Writeln(OutFile,V0Str);
        Writeln(OutFile,V1Str);
        Writeln(OutFile,V2Str);
        Writeln(OutFile,V3Str);
        Writeln(OutFile,V4Str);
    End;
  End;  (* While Not EOF *)
  Close(InFile);

  Flush(OutFile);
  Close(OutFile);
  TempCom:='FILE FORMATTED FOR REGRESSION ANALYSIS USING INPUT FILE ' + IFileNa
  UpdateCom;

End.^Z
```

Appendix J.20

GetDirectory

```
Procedure GetDirectory(TmpMask:Char12arr);

VAR

  DTA:                array [1..43] of byte;
  Mask:               Char12arr;
  Reg:                RegRec;
  TheDate,Tim:        integer;
  CheckSize:          array [1..4] of byte;
  CSize:              real;
  J,Error,I:          integer;
  CH:                 Char;
  NamR:               String20;

 (*==================================================================*)


Begin

  If ChkDir <> ProgDrvDir Then
     ChDir(ChkDir);
  FillChar(DTA,SizeOf(DTA),0);
  FillChar(Mask,SizeOf(Mask),0);
  FillChar(NamR,SizeOf(NamR),0);
  NoFiles:=0;

  Reg.ax:=$1A00;
  Reg.ds:=Seg(DTA);
  Reg.dx:=Ofs(DTA);
  Intr($21,Reg);
  Mask:=TmpMask;
  Reg.ax:=$4E00;
  Reg.ds:=Seg(Mask);
  Reg.dx:=Ofs(Mask);
  Reg.cx:=22;
  Intr($21,Reg);
  Error:=0;
  Error:=Reg.ax and $FF;
  FirstRec:=Nil;
  If (Error = 0) then Begin
    I:=1;
    Repeat
      NamR[I]:=Chr(Mem[Seg(DTA):Ofs(DTA)+29+I]);
```

```
  I:=I + 1;
Until not (NamR[I - 1] in [' '..'~']) or (I > 20);
NamR[0]:=Chr(I - 1);

For I:=1 to 4 Do
  CheckSize[I]:=Mem[Seg(DTA):Ofs(DTA)+25 + I];

Tim:=MemW[Seg(Dta):Ofs(DTA)+22];

TheDate:=MemW[Seg(Dta):Ofs(DTA)+24];

New(NewRec);
NewRec^.Name:=NamR;

CSize:=0;
CSize:=CSize + CheckSize[1];
CSize:=CSize + CheckSize[2] * 256.0;
CSize:=CSize + CheckSize[3] * 65536.0;
If CheckSize[4] > 0 Then
  For I:=1 to CheckSize[4] Do
       CSize:=CSize * 10;
NewRec^.Size:=CSize;

If Tim < 0 Then
   Tim:=1024 + ((32767 + Tim) DIV 32)
Else Tim:=Tim DIV 32;
NewRec^.Minu:=Tim Mod 64;
Tim:=Tim DIV 64;
NewRec^.Hr:=Tim;

NewRec^.Day:=TheDate MOD 32;
TheDate:=TheDate DIV 32;
NewRec^.Month:=TheDate MOD 16;
TheDate:=TheDate DIV 16;
NewRec^.Year:=TheDate + 80;

If FirstRec = Nil Then Begin
   FirstRec:=NewRec;
   NewRec^.Prev:=Nil;
End
Else Begin
   LastRec^.Next:=NewRec;
   NewRec^.Prev:=LastRec;
End;
LastRec:=NewRec;
LastRec^.next:=Nil;
NoFiles:=NoFiles + 1;
```

```
End;  (* Error = 0 *)

While (Error = 0) do Begin
   Error:=0;
   Reg.ax:=$4F00;
   Reg.cx:=22;
   Intr($21,Reg);
   Error:=Reg.ax and $FF;
   I:=1;
   Repeat
     NamR[I]:=Chr(Mem[Seg(DTA):Ofs(DTA)+29+I]);
     I:=I + 1;
   Until not (NamR[I - 1] in [' '..'~']) or (I > 20);
   NamR[0]:=Chr(I - 1);

   For I:=1 to 4 Do
     CheckSize[I]:=Mem[Seg(DTA):Ofs(DTA)+25 + I];

   Tim:=MemW[Seg(Dta):Ofs(DTA)+22];

   TheDate:=MemW[Seg(Dta):Ofs(DTA)+24];

   If Error = 0 Then Begin
      New(NewRec);
      NewRec^.Name:=NamR;

      CSize:=0;
      CSize:=CSize + CheckSize[1];
      CSize:=CSize + CheckSize[2] * 256.0;
      CSize:=CSize + CheckSize[3] * 65536.0;
      If CheckSize[4] > 0 Then
      For I:=1 to CheckSize[4] Do
          CSize:=CSize * 10;
      NewRec^.Size:=CSize;

      If Tim < 0 Then
         Tim:=1024 + ((32767 + Tim) DIV 32)
      Else Tim:=Tim DIV 32;
      NewRec^.Minu:=Tim Mod 64;
      Tim:=Tim DIV 64;
      NewRec^.Hr:=Tim;

      NewRec^.Day:=TheDate MOD 32;
      TheDate:=TheDate DIV 32;
      NewRec^.Month:=TheDate MOD 16;
      TheDate:=TheDate DIV 16;
      NewRec^.Year:=TheDate + 80;
```

```
            If FirstRec = Nil Then Begin
               FirstRec:=NewRec;
               NewRec^.Prev:=Nil;
            End
            Else Begin
               LastRec^.Next:=NewRec;
               NewRec^.Prev:=LastRec;
            End;
            LastRec:=NewRec;
            LastRec^.next:=Nil;
         End;
         If Error = 0 Then NoFiles:=NoFiles + 1;
      End;  (* While Error = 0 *)
      If ChkDir <> ProgDrvDir Then
         ChDir(ProgDrvDir);

   End;  (* Procedure GetDir *)
^Z
```

Appendix J.21

Help

```
Procedure Help(HelpMsg1,HelpMsg2:String255);

VAR
  Ch: char;
  Image:ImageType;

  Begin

    If CrtMode = 7 Then
       Image:=MonoBuffer
    Else
       Image:=ColorBuffer;
    BoxWin(10,10,70,20,' Press Space Bar to Exit ');
    TextColor(Black);
    TextBackGround(LightGray);
    ClrScr;
    GotoXY(1,2);
    Writeln(HelpMsg1);
    GotoXY(1,WhereY);
    Writeln(HelpMsg2);
    Ch:=' ';
    Read(kbd,ch);
    If CrtMode = 7 Then
       MonoBuffer:=Image
    Else ColorBuffer:=Image;
    TextColor(Yellow);
    TextBackGround(Blue);
    Window(2,2,79,24);

  End;^Z
```

[BLANK PAGE]

Appendix J.22

Include710

```
Program Include710(input,output);
{$C-}
```

Deborah A. Holder

```
LABEL
   TRYAGAIN;

TYPE
   Array70 = Array [1..70] of char;
   String255 = string[255];
   String70 = String[70];
   String50 = String[50];
   String20 = String[20];
   String10 = String[10];
   String3  = String[3];
   String150 = string[150];
   Char12arr = array [1..12] of char;
   HoldName = array [1..25] of char;
   RegRec = record
              ax,bx,cx,dx,bp,si,di,ds,es,flags: integer;
            end;

   DPointer = ^DirRec;
   DirRec = record
              Size: real;
              Hr,Minu: integer;
              Year,Month,Day: integer;
              Next,Prev: DPointer;
              Name: String20;
            End;

   SqrRec = record
               SqrLine:Array70;
              end;

   SqrString = string[10];
   Path = String[64];
   ImageType = array [1..4096] of char;
```

```
   DateTimeType = String[8];

VAR
   Sqrdqt: File of SqrRec;
   OFileName,SFileName,IFileName: Path;
   GFN,INFile,OutFile,MtbfFile: text;
   ILine,OLine: String255;
   T1Str,T2Str,T3Str: String10;
   T2,T3,Delta,Max: Integer;
   A1Str,A2Str,A3Str,A4Str,A5Str: String10;
   A1,A2,A3,A4,A5:Real;
   Sx,S1,S2,S3,S4,S5:Real;
   Rs1,CCO,F: real;
   HeapTop: ^Integer;
   NoFiles: integer;
   Top,Bottom: DPointer;
   FirstRec,LastRec,NewRec: DPointer;
   FileEnd,I,J,K: integer;
   NextSq,Sq,Ifrs,Code: integer;
   FN: array [1..50] of real;
   SqrIn: SqrRec;
   ChkVal,HoldVal:integer;
   UppCase,First: Boolean;
   ChkLine: String70;
   SqStrArr: array [1..50] of SqrString;
   SqAdd: array [1..50] of integer;
   ChkSqd: SqrString;
   CpStr: SqrString;
   SqrFileName,CurDir: Path;
   Ans,NewDrive: char;
   Default,Quit,Found,ReqFile,ValFile: Boolean;
   Replace,Initial: integer;
   Field,MaxField: Integer;
   Line: String150;
   PrtDate,PrtTime: DateTimeType;
   Image: ImageType;
   CrtMode: Byte absolute $0040:$0049;
   MonoBuffer: ImageType absolute $B000:$0000;
   ColorBuffer: ImageType absolute $B800:$0000;
   ChkFile,ChkDir,ProgDrvDir,DrvDir: Path;
   TempCom: String255;
   RdStr,TmpCom1,TmpCom2,TmpCom3,TmpCom4,TmpCom5: Path;
   HlpMsg1,HlpMsg2: String255;

{$IBoxWin.pas}
{$IHelp.pas}
{$IAddDir.pas}
```

```
{$IComments.pas}
{$IExists.pas}
{$IDiskSpace.pas}
{$IGetDir.pas}
{$IDirlst.pas}
{$IFileName.pas}
{$IAddFileExt.pas}
{$ITime.pas}
{$IDate.pas}
{$IReadField.pas}
{$IReadCom.pas}
{$IDisCom.pas}
{$IUpdateCom.pas}

(*=================================================================*)
   Procedure BuildStr;

   VAR
     FirstLine,SecondLine: String150;
     S1Str,S2Str,S3Str,S4Str,S5Str: String10;

   Begin  (* Procedure BuildStr *)

     S2:=Int(S2 + 0.5);
     S3:=Int(S3 + 0.5);
     S4:=Int(S4 + 0.5);
     S5:=Int(S5 + 0.5);

     FirstLine:=Copy(ILine,1,113);
     SecondLine:=Copy(ILine,126,11);
     Str(S1:7:0,S1Str);
     Str(S2:6:0,S2Str);
     Str(S3:6:0,S3Str);
     Str(S4:6:0,S4Str);
     Str(S5:6:0,S5Str);
     OLine:='';
     OLine:=FirstLine + SecondLine + S1Str + S2Str + S3Str + S4Str + S5Str;

   End;  (* Procedure BuildStr *)
(*=================================================================*)
   Function ValueStr(InStr:String10):Real;

   VAR
     I: Integer;
     V: Real;
```

```
  Begin  (* Function ValueStr *)

    I:=1;
    Repeat
      If InStr[I] = ' ' Then
         Delete(InStr,I,1)
      Else I:=I + 1;
    Until (I >= Length(InStr));
    Val(InStr,V,Code);
    ValueStr:=V;

  End;  (* Function ValueStr *)
(*=====================================================================*)

Begin  (* Program Include710 *)

  GraphBackGround(Blue);
  TextBackGround(Blue);
  TextColor(Yellow);
  ClrScr; Replace:=0;
  BoxWin(1,1,80,24,'  STEP 3  ');
  GotoXY(26,10);
  Write('>>> Include R0710 Data <<<');
  GotoXY(1,1);
  SqrFileName:='SqrDqt.sqr';
  GetDir(0,ProgDrvDir);
  ChkDir:=''; ChkFile:=''; DrvDir:='';
  Default:=False;
  Assign(GFN,'GFN_Nail');
  Reset(GFN);
  Readln(GFN,DrvDir);
  Readln(GFN,ChkFile);
  Close(GFN);
  ChkDir:=DrvDir;
  If Length(ChkDir) = 2 Then ChkDir:=ChkDir + '\';
  Delay(1000);
  If ChkFile <> '**' Then Default:=True;
  If Default = False Then Begin
     ClrScr;
     GotoXY(10,5);
     Write('This program requires an Input file named Sqrdqt.sqr.  This file');
     GotoXY(10,6);
     Write('is assumed to reside on the ',DrvDir[1],' Drive.  If this drive does not');
     GotoXY(10,7);
     Write('contain the required file, input the correct drive: _');
     GotoXY(WhereX -1, WhereY);
     Readln(trm,NewDrive);
```

```
    NewDrive:=Upcase(NewDrive);
    If Ord(NewDrive) <> 13 Then
       SqrFileName:=NewDrive + ':' + SqrFileName
    Else If ProgDrvDir <> DrvDir Then AddDir(SqrFileName);
 End
 Else If ProgDrvDir <> DrvDir Then AddDir(SqrFileName);
 SFileName:='';
 SFileName:=SqrFileName;
 ValFile:=Exists(SFileName);
 If ValFile = True Then Begin
    Assign(Sqrdqt,SFileName);
    Reset(Sqrdqt);
    FileEnd:=FileSize(Sqrdqt) - 1;
    IFileName:='';
    ClrScr;
    Ifrs:=0;
    If Default = False Then Begin
       GotoXY(5,10); Write('Do you want to set zero Tbf = adv.? [Y,N]: ');
       RdStr:='';
       HlpMsg1:=' Enter  Y if you want to set zero Tbf = adv.';
       HlpMsg2:='';
       Field:=1;  MaxField:=1;  Quit:=False;
       ReadField(RdStr,WhereX,WhereY,1);
       If RdStr <> '' Then Ans:=UpCase(RdStr[1])
       Else Ans:='N';
       If Ans = 'Y' Then Ifrs:=1;
    End;
    ClrScr;
    (* Load Mtbf.fac into array FC *)
    For I:=1 to 50 Do FN[I]:=0;
    I:=1;
    ValFile:=Exists('Mtbf.fac');
    If ValFile = True Then Begin
      Assign(MtbfFile,'Mtbf.Fac');
      Reset(MtbfFile);
      While ((NOT EOF(MtbfFile)) and (I <=48)) Do Begin
         Readln(MtbfFile,FN[I]);
         I:=I + 1;
      End;
      Close(MtbfFile);
    End;

    (* Load Squad Data into arrays *)
    For I:=1 to 50 Do Begin
        SqStrArr[I]:='';
        SqAdd[I]:=0;
    End;
```

```
Seek(Sqrdqt,0);
Read(Sqrdqt,Sqrin);
ChkLine:='';
ChkLine:=Sqrin.SqrLine;
CpStr:=Copy(ChkLine,1,4);
If CpStr[1] = ' ' Then Delete(CpStr,1,1);
Val(CpStr,HoldVal,Code); Sq:=0;
For I:=1 to FileEnd Do Begin
   Seek(Sqrdqt,I);
   Read(Sqrdqt,SqrIn);
   ChkLine:='';
   ChkLine:=SqrIn.SqrLine;
   If ((Pos('ORG :',ChkLine) <> 0) and (Sq < 50)) Then Begin
      Sq:=Sq + 1;
      CpStr:=Copy(ChkLine,6,3);
      SqStrArr[Sq]:=CpStr;
      SqAdd[Sq]:=I;
   End;
End;

If ((ChkFile <> '**') and (Pos('DTB',ChkFile) > 0)) Then Begin
   IFileName:=ChkFile;
   If DrvDir <> ProgDrvDir Then AddDir(IFileName);
End
Else Begin
   Mark(HeapTop);
   GetDirectory('????????.DTB');
   ReqFile:=True;
   ValFile:=False;
   IFileName:='';
   While ValFile = False Do Begin
       ReqFile:=True;
       IFileName:='                    ';
       Filename(IFileName,' Input   ');
       AddFileExt(IFileName,'DTB');
       If DrvDir <> ProgDrvDir Then AddDir(IFileName);
       ValFile:=Exists(IFileName);
       If ValFile = False Then Begin
          GotoXY(1,12); ClrEOL;
          GotoXY(15,12); Write(^G,'File Does not Exist !!');
          Delay(3000);
       End;
   End;
   Release(HeapTop);
End;

Assign(InFile,IFileName);
Reset(InFile);
```

```
If ((ChkFile <> '**') and (Pos('DTB',ChkFile) > 0)) Then Begin
   OFileName:=ChkFile;
   AddFileExt(OFileName,'DTC');
   ChkFile:=OFileName;
   If ProgDrvDir <> DrvDir Then AddDir(OFileName);
   Assign(GFN,'GFN_Nail');
   ReWrite(GFN);
   Writeln(GFN,DrvDir);
   Writeln(GFN,ChkFile);
   Close(GFN);
End
Else Begin
   Mark(HeapTop);
   GetDirectory('????????.dtc');
   ValFile:=True;
   OFileName:='';
   While ValFile = True Do Begin
       ReqFile:=True;
       OFileName:='                    ';
       Filename(OFileName,'  Output  ');
       AddFileExt(OFileName,'DTC');
       If DrvDir <> ProgDrvDir Then AddDir(OFileName);
       ValFile:=Exists(OFileName);
       If ValFile = True Then Begin
          GotoXY(1,12); ClrEOL;
          GotoXY(15,12); Write(^G,'File Already Exist !!');
          GotoXY(15,13); Write('Do you wish to overwrite?: _');
          GotoXY(WhereX -1,WhereY);
          Read(Kbd,Ans);
          Ans:=UpCase(Ans);
          GotoXY(WhereX -1,WhereY);
          Write(Ans);
          If Ans = 'Y' Then ValFile:=False;
       End;
   End;
   Release(HeapTop);
End;
Assign(OutFile,OFileName);
Rewrite(OutFile);
ClrScr;
GotoXY(30,10);
Write('INCLUDE R710 DATA');
(* Build DTC File *)
rs1:=0;  Max:=512;
While NOT EOF(InFile) Do Begin
   A1:=0;  A2:=0;  A3:=0;  A4:=0;  A5:=0;
   S1:=0;  S2:=0;  S3:=0;  S4:=0;  S5:=0;
```

```
Readln(InFile,ILine);
T1Str:=Copy(ILine,53,3);
T2Str:=Copy(ILine,56,6);
T3Str:=Copy(ILine,129,6);
T2:=Trunc(ValueStr(T2Str));
T3:=Trunc(ValueStr(T3Str));
Delta:=T2 - T3;
If Delta >= 900 Then Begin
   T3:=T3*10 + 5;
   If T3 < T2 Then T3:=T2;
End;
Found:=False;  J:=1;
While ((Found = False) and (J <= Sq)) Do Begin
    If SqStrArr[J] = T1Str Then Found:=True
    Else J:=J + 1;
End;
If J + 1 <=Sq Then NextSq:=SqAdd[J + 1] - 1
Else NextSq:=FileEnd;
Found:=False;  I:=SqAdd[J] + 1;
While ((Found = False) and (I <=NextSq)) Do Begin
     Seek(Sqrdqt,I);
     Read(Sqrdqt,SqrIn);
     ChkLine:='';
     ChkLine:=SqrIn.SqrLine;
     A1Str:=Copy(ChkLine,1,6);
     A2Str:=Copy(ChkLine,6,5);
     A3Str:=Copy(ChkLine,12,6);
     A4Str:=Copy(ChkLine,18,6);
     A5Str:=Copy(ChkLine,24,6);
     A1:=ValueStr(A1Str);
     A2:=ValueStr(A2Str);
     A3:=ValueStr(A3Str);
     A4:=ValueStr(A4Str);
     A5:=ValueStr(A5Str);
     If (T3 - A1) <=30 Then Found:=True
     Else I:=I + 1;
End;
If Found = False Then Begin
   A1:=0;  A2:=0;  A3:=0;  A4:=0;  A5:=0;
   S1:=0;  S2:=0;  S3:=0;  S4:=0;  S5:=0;
End;
F:=1 - (T3 Mod 30) /30.0;
S2:=A2 * F;  S3:=A3 * F;  S4:=A4 * F;  S5:=A5 * F;
CCO:=F;
Found:=False;
While ((Found = False) and (I <=NextSq)) Do Begin
```

```
            CCO:=CCO + 1;
            Seek(Sqrdqt,I);
            Read(Sqrdqt,SqrIn);
            ChkLine:='';
            ChkLine:=SqrIn.SqrLine;
            A1Str:=Copy(ChkLine,1,6);
            A2Str:=Copy(ChkLine,6,5);
            A3Str:=Copy(ChkLine,12,6);
            A4Str:=Copy(ChkLine,18,6);
            A5Str:=Copy(ChkLine,24,6);
            A1:=ValueStr(A1Str);
            A2:=ValueStr(A2Str);
            A3:=ValueStr(A3Str);
            A4:=ValueStr(A4Str);
            A5:=ValueStr(A5Str);
            If (T2 - A1) <=30 Then Found:=True
            Else Begin
               I:=I + 1;
               S2:=S2 + A2;  S3:=S3 + A3;  S4:=S4 + A4;  S5:=S5 + A5;
            End;
      End;
      If Found = False Then Begin
         If CCO = 0 Then CCO:=1;
         S2:=S2/CCO;  S3:=S3/CCO;  S4:=S4/CCO; S5:=S5/CCO;
      End
      Else Begin
         F:=(T2 Mod 30)/30.0;
         CCO:=CCO + F -1;
         If CCO <=0 Then CCO:=1;
         S2:=(S2 + A2 * F)/CCO;
         S3:=(S3 + A3 * F)/CCO;
         S4:=(S4 + A4 * F)/CCO;
         S5:=(S5 + A5 * F)/CCO;
         S1:=T2 - T3;
         If T3 <= 0 Then Begin
            S1:=Trunc(T2 * 12/365) + 1;  If S1 > 50 Then S1:=50;
            SX:=(Max - T2) * Random;     If Sx < 0 Then Sx:=0;
            S1:=FN[Trunc(S1)] * (Sx + T2);
            S1:=Trunc(S1 + 0.5);
         End;
         If S1 < 0 Then S1:=trunc(Rs1 * 100)/100;
         If ifrs = 1 Then Rs1:=(Rs1 * 1 + S1)/1;
      End;
      BuildStr;
      Writeln(OutFile,OLine);
   End;  (* While NOT EOF *)
   Close(InFile);
```

```
      Close(OutFile);
      TempCom:='FILE, ' + IFileName + ', HAS BEEN COMBINED WITH THE OPERATIONAL
DATA FROM THE R710 REPORT';
      UpdateCom;
   End
   Else Begin
      ClrScr;
      GotoXY(15,12);
      Write(^G,'Sqrdqt.sqr does not exist');
      GotoXY(15,13);
      Write('You Need to Execute Step 13 First');
      Delay(3000);
   End;

End.    (* Program Include710 *)^Z
```

Appendix J.23

LastFail

```
Program LastFail(Input,Output);
{$C-}
```

STEP 2 - Evaluate Time between Failures. This step simplifies the data for analysis. The aircraft tail numbers are used to find successive failure dates. When dates are identified the difference is used to calculate the time between failures (TBF).

Deborah A. Holder

```
TYPE
  HoldName = array [1..25] of char;
  RegRec = record
             ax,bx,cx,dx,bp,si,di,ds,es,flags: integer;
           end;
  ArrReal = Array [1..100,1..2] of real;
  String255 = string[255];
  String128 = String[128];
  String3   = String[3];
  String20  = String[20];
  String50  = String[50];
  TempRec   = Record
                TempVar: String128;
              End;
  Char12arr = array [1..12] of char;
  String6 = String[6];
  String10 = String[10];
  DPointer = ^DirRec;
  DirRec = record
             Size: real;
             Hr,Minu: integer;
             Year,Month,Day: integer;
             Next,Prev: DPointer;
             Name: String20;
           end;
  ImageType = array [1..4096] of char;
  Path = string[64];
```

```
VAR
  TN: ArrReal;
  HeapTop: ^Integer;
  NoTN,NoFiles: integer;
  Top,Bottom: DPointer;
  FirstRec,LastRec,NewRec: DPointer;
  TName,OFileName,IFileName: Path;
  TailFile,GFN,INFile,OutFile: Text;
  TempFile: File of TempRec;
  LineRec: TempRec;
  TempLine,Line: String128;
  AStr: Array [1..20] of String20;
  HoldSt,TTT,T,TSame: String20;
  SD,SAD1,SDM: String20;
  SaveSub,Row,FRST,SizeTTT: Integer;
  L,K,Code,Y,J,I,CSame: integer;
  CX,V,BN,BOT: real;
  FRec,LS: integer;
  Ans: Char;
  UppCase,Default,Quit,ReqFile,ValFile: Boolean;
  Field,MaxField: Integer;
  Image: ImageType;
  CrtMode: Byte absolute $0040:$0049;
  MonoBuffer: ImageType absolute $B000:$0000;
  ColorBuffer: ImageType absolute $B800:$0000;
  ChkFile,ChkDir,ProgDrvDir,DrvDir: Path;
  TempCom: String255;
  TmpCom1,TmpCom2,TmpCom3,TmpCom4,TmpCom5: Path;
  HlpMsg1,HlpMsg2: String255;


{$IBoxWin.pas}
{$IHelp.pas}
{$IAddDir.pas}
{$IComments.pas}
{$IExists.pas}
{$IDiskSpace.pas}
{$IGetDir.pas}
{$IDirLst.pas}
{$IFileName.pas}
{$IAddFileExt.pas}
{$IReadField.pas}
{$IReadCom.pas}
{$IDisCom.pas}
{$IUpdateCom.pas}
```

```
Procedure InitTN;
(*-----------------------------------------------------------------*)

 Begin (* Procedure InitTN *)

    For I:=1 to 100 Do Begin
        TN[I,1]:=0;  TN[I,2]:=0;
    End;  NoTN:=0;

    IFileName:=''; IFileName:='Tail.num';
    If DrvDir <> ProgDrvDir Then AddDir(IFilename);
    If Exists(IFileName) = True Then Begin
       Assign(TailFile,IFileName);
       ReSet(TailFile);
       While NOT EOF(TailFile) Do Begin
          NoTN:=NoTN + 1;
          Readln(TailFile,TN[NoTN,1],TN[NoTn,2]);
       End;
       Close(TailFile);
    End;


 End;  (* Procedure InitTN *)
(*-----------------------------------------------------------------*)

Procedure BreakUpStr;
(*-----------------------------------------------------------------*)

 Begin (* Procedure BreakUpStr *)

    AStr[1]:=Copy(Line,1,1);
    AStr[2]:=Copy(Line,2,7);
    AStr[3]:=Copy(Line,9,11);
    AStr[4]:=Copy(Line,20,16);
    AStr[5]:=Copy(Line,36,6);
    AStr[6]:=Copy(Line,42,11);
    AStr[7]:=Copy(Line,53,5);
    AStr[8]:=Copy(Line,58,5);
    AStr[9]:=Copy(Line,63,9);
    AStr[10]:=Copy(Line,72,5);
    AStr[11]:=Copy(Line,77,9);
    AStr[12]:=Copy(Line,86,2);
    AStr[13]:=Copy(Line,88,2);
    AStr[14]:=Copy(Line,90,2);
    AStr[15]:=Copy(Line,92,4);
    AStr[16]:=Copy(Line,96,4);
```

```
      AStr[17]:=Copy(Line,100,8);
      AStr[18]:=Copy(Line,108,20);
      AStr[19]:=Copy(Line,126,1);

   End;  (* Procedure BreakUpStr *)
  (*-----------------------------------------------------------------*)


Begin

 GraphBackGround(Blue);
 TextBackGround(Blue);
 TextColor(Yellow);
 ClrScr; Default:=False;
 BoxWin(1,1,80,24,'  STEP 2  ');
 GotoXY(20,10);
 Write('>>> Evalute Time Between Failures <<<');
 GotoXY(1,1);
 GetDir(0,ProgDrvDir);
 Assign(GFN,'GFN_Nail');
 ReSet(GFN);
 ChkDir:=''; ChkFile:='';
 Readln(GFN,DrvDir);
 Readln(GFN,ChkFile);
 Close(GFN);
 ChkDir:=DrvDir;
 If Length(ChkDir) = 2 Then ChkDir:=ChkDir + '\';
 InitTN;
 Delay(1000);
 ReqFile:=True;
  (* Assign Input File and OutPut File *)


  If ((ChkFile <> '**') and (Pos('DTA',ChkFile) > 0)) Then Begin
     Default:=True;
     IFileName:=''; IFileName:=ChkFile;
     If DrvDir <> ProgDrvDir Then AddDir(IFilename);
  End
  Else Begin
     Mark(HeapTop);
     GetDirectory('????????.DTA');
     ValFile:=False; IFileName:='';
     While ValFile = False Do Begin
        IFileName:='                    ';
        FileName(IFileName,'  Input  ');
        AddFileExt(IFileName,'DTA');
        If DrvDir <> ProgDrvDir Then AddDir(IFilename);
```

```
      ValFile:=Exists(IFileName);
      If ValFile = False Then Begin
         GotoXY(1,12); ClrEOL;
         GotoXY(15,12); Write(^G,'File Does Not Exist !!');
         Delay(3000);
      End;
   End;
   Release(HeapTop);
End;
If ((ChkFile <> '**') and (Pos('DTA',ChkFile) > 0)) Then Begin
   OFileName:=''; OFileName:=ChkFile;
   AddFileExt(OFileName,'DTB');
   ChkFile:=OFileName;
   Assign(GFN,'GFN_Nail');
   ReWrite(GFN);
   Writeln(GFN,DrvDir);
   Writeln(GFN,ChkFile);
   Close(GFN);
   If DrvDir <> ProgDrvDir Then AddDir(OFilename);
End
Else Begin
   Mark(HeapTop);
   GetDirectory('????????.DTB');
   ReqFile:=True;
   ValFile:=True; OFileName:='';
   Repeat
      OFileName:='                    ';
      FileName(OFileName,' Output ');
      AddFileExt(OFileName,'DTB');
      If DrvDir <> ProgDrvDir Then AddDir(OFilename);
      ValFile:=Exists(OFileName);
      If ValFile = True Then Begin
         GotoXY(1,12); ClrEOL;
         GotoXY(1,13); ClrEOL;
         GotoXY(17,12); Write(^G,'File Already Exist !!');
         GotoXY(17,13); Write('Do you wish to overwrite ?: _');
         GotoXY(WhereX - 1,WHereY);
         Read(Kbd,Ans);
         Ans:=UpCase(Ans);
         GotoXY(WhereX,WhereY);
         Write(Ans);
         If Ans = 'Y' Then ValFile:=False;
      End;
   Until (ValFile = False);
   Release(HeapTop);
End;
```

```
Assign(InFile,IFileName);
Reset(InFile);
Assign(OutFile,OFileName);
ReWrite(OutFile);
TName:='Temp.zzz';
If DrvDir <> ProgDrvDir Then AddDir(TName);
Assign(TempFile,TName);
ReWrite(TempFile);

ClrScr;
GotoXY(20,10);
Write('CALCULATING TIME BETWEEN FAILURES');


While NOT EOF(InFile) Do Begin
   Readln(InFile,Line);
   BreakUpStr;
   TTT:=AStr[19];  SizeTTT:=Pos(TTT,'G');
   T:=AStr[2];
   If ((T = '') or (T = 'R0520 P') or (T = 'ODE NUM') or (T = '        ')
       or (SizeTTT > 0))
   Then
   Else Begin
       LineRec.TempVar:=Line;
       Seek(TempFile,FileSize(TempFile));
       Write(TempFile,LineRec);
   End;

End;  (* While Not EOF InFile  *)
Close(InFile);

J:=-1;  CSame:=-1;  TSame:='';  FRec:=0;
For I:=0 to FileSize(TempFile) - 1 Do Begin
    Seek(TempFile,I);
    Read(TempFile,LineRec);
    Line:=LineRec.TempVar;
    T:=Copy(Line,2,6);
    CSame:=CSame + 1;
    If T <> TSame Then Begin
      CSame:=0;
      FRec:=I -1;
    End;
    TSame:=T;
    CX:=0;  SDM:='';  FRST:=0; FRec:=0;
    If FRec <> -1 Then
    While (FRec < J) Do Begin
```

```
    Seek(TempFile,FRec);
    Read(TempFile,LineRec);
    TempLine:=LineRec.TempVar;
    If T = Copy(TempLine,2,6) Then Begin
       CX:=CX + 1;
       SD:=Copy(TempLine,58,5);
       SAD1:=Copy(TempLine,72,5);
       SDM:=SD;
       If SDM < SAD1 Then SDM:= SAD1;
    End;
    FRec:=FRec + 1;
End;
If SDM = '' Then V:=0
Else Val(SDM,V,Code);
If V = 0 Then Begin
   HoldSt:=Copy(Line,2,7);
   Val(HoldSt,BN,Code);
   Bot:=0;
   For Row:=1 to NoTN Do Begin
       If ((BN >= TN[Row,1]) and (BN <=TN[Row,2])) Then Begin
          Bot:= TN[Row,2] - TN[Row,1];
          SaveSub:=Row;
       End;
   End;
   If Bot <> 0 Then Begin
       V:=365.0 * (SaveSub - 1 + (BN - TN[SaveSub,1]))/Bot;
       Str(V:6:0,SDM);
       FRST:=1;
   End;
End;

Ls:=Length(SDM);
If LS < 6 Then
   For K:=1 to 6 - LS Do Insert(' ',SDM,1);
If FRST = 1 Then
   CX:=CX + 0.1;
Write(OutFile,Line,SDM);
If Frac(CX) = 0 Then
  Str(CX:2:0,HoldSt)
Else Begin
  Str(CX:3:1,HoldSt);
  K:=Pos('.',HoldSt);
  Delete(HoldSt,1,K-1);
  For L:=1 to K - 1 Do Insert(' ',HoldSt,1);
End;
Writeln(OutFile,HoldSt);
J:=I - CSame;
```

```
    End;  (* For Loop *)

    Flush(OutFile);
    Close(OutFile);
    Close(TempFile);
    Erase(TempFile);
    TempCom:='CALCULATE LAST FAILURE USING INPUT FILE ' + IFilename;
    UpDateCom;

End.^Z
```

Appendix J.24

MIR

```
Program MIR(Input,Output);
{$C-}
```

STEP 16 - MIR Study. This analysis is used to project the number of removals and repairs at the O, I, and D levels using a data set created by the Step 2 activity.

Deborah A. Holder

```
CONST
  nmax = 600;

TYPE
  HoldName = array [1..25] of char;
  RegRec = record
             ax,bx,cx,dx,bp,si,di,ds,es,flags: integer;
           end;
  String255 = String[255];
  String140 = String[140];
  String50  = String[50];
  String20  = String[20];
  String3   = String[3];
  Char12arr = array [1..12] of char;
  String6 = String[6];
  String10 = String[10];
  DPointer = ^DirRec;
  DirRec = record
             Size: real;
             Hr,Minu: integer;
             Year,Month,Day: integer;
             Next,Prev: DPointer;
             Name: String20;
           end;
  SRec = record
           SLine: Array [1..70] of char;
         end;
  DateTimeType = String[8];
  ImageType = array[1..4096] of char;
  Path = string[64];
```

```
VAR
  HeapTop: ^Integer;
  Code,J,K,I,NoFiles: integer;
  Top,Bottom: DPointer;
  FirstRec,LastRec,NewRec: DPointer;
  OFileName,IFileName,SFileName: Path;
  GFN,PrtFile,INFile: Text;
  SFile: File of SRec;
  SqdRec: SRec;
  SqdLine: String[70];
  BYStr: String3;
  ChkMonth,PrtYr,BY: Integer;
  PrtMonth: String6;
  PrtY1,PrtY2: Integer;
  PrtM1,PrtM2: String6;
  Ans: char;
  Rw,Cl: integer;
  Line: String140;
  UppCase,ReqFile,ValFile: Boolean;
  PrtOut,SquProj: char;
  SquadCode: String6;
  ProjCon: integer;
  NoMonths,NoYrs: integer;
  CHkStr,ProjConStr,NoMonthsStr,NoYrsStr: String6;
  Ch: char;
  PrtTime,PrtDate: DateTimeType;
  Wuc,bn,cn,tsq,TStr,Tato,Tati,Tatd,NpStr: String10;
  Vc,FStr: String10;
  T,T1,T2,Dt,np,F: real;
  ii,nobs,nlst: integer;
  ni,n1,n2: integer;
  ap,noi,noo,nod,nro: integer;
  Proj,cnp,ndod,nodg: integer;
  fo,fi,fd: real;
  vMax,mtbf,ddt,ta: real;
  A,B,Bot1,Top1,afsd: real;
  sd,advc,advn,nc,dn,S1,S2,S3,S4,Sd2: real;
  SqrRootTwelve,sdx,ndl: real;
  n: array [1..600] of integer;
  Prjo,Prji,Prjd: array [1..50] of real;
  TotPrjo,TotPrji,TotPrjd: real;
  tt,sxd,nct,nco,nci,ncd,sno,sni,snd:real;
  icm,icy: integer;
  Field,MaxField: Integer;
  Default,Quit: Boolean;
  Image: ImageType;
  CrtMode: Byte absolute $0040:$0049;
```

```
  MonoBuffer: ImageType absolute $B000:$0000;
  ColorBuffer: ImageType absolute $B800:$0000;
  ChkDir,ChkFile,ProgDrvDir,DrvDir: Path;
  TempCom: String255;
  TmpCom1,TmpCom2,TmpCom3,TmpCom4,TmpCom5: Path;
  HlpMsg1,HlpMsg2: String255;


{$IBoxWin.pas}
{$IHelp.pas}
{$IAddDir.pas}
{$IComments.pas}
{$IExists.pas}
{$IDiskSpace.pas}
{$IGetDir.pas}
{$IDirLst.pas}
{$IFileName.pas}
{$IAddFileExt.pas}
{$ITime.pas}
{$IDate.pas}
{$IPrtComments.pas}
{$IReadField.pas}
{$IReadCom.pas}
{$IDisCom.pas}
{$IUpdateCom.pas}

(*=========================================================================*)
 Procedure InputStr(Var InStr:String6; Limit:integer; Col,Row:integer; Num:Boolean);

  VAR
    Ch: char;
    I: integer;

 Begin  (* Procedure InputStr *)

  I:=1;
  InStr:='';
  Repeat
     GotoXY(Col,Row);
     Read(Kbd,Ch);
     Case Ord(ch) of
                27: Begin
                      Read(Kbd,ch);
                      If ord(ch) = 59 Then Help(HlpMsg1,HlpMsg2);
                    End;
   33..47,58..125: If ((I <= Limit) and (Num = False)) Then Begin
                       Ch:=UpCase(Ch);
```

```
                    Insert(Ch,InStr,I);
                    GotoXY(Col,Row);
                    Write(Ch);
                    I:=I + 1;
                    Col:=Col + 1;
                  End;
          48..57: If I <= Limit Then Begin
                    Ch:=UpCase(Ch);
                    Insert(Ch,InStr,I);
                    GotoXY(Col,Row);
                    Write(Ch);
                    I:=I + 1;
                    Col:=Col + 1;
                  End;
                8: If I > 1 Then Begin
                    I:=I - 1;
                    Col:=Col - 1;
                    Delete(InStr,I,1);
                    GotoXY(Col,Row);
                    Write('_');
                  End;
    End; (* Case Statement *)
  Until (Ord(ch) = 13);

End;
(*=========================================================================*)
Procedure FindAandB;

VAR
  I: integer;

Begin  (* Procedure FindAandB *)
  S1:=0;  s2:=0;  s3:=0;  s4:=0;  cnp:=0;

  For I:=nlst to nobs Do Begin
      cnp:=cnp + 1;
      S1:=S1 + n[I];
      S2:=S2 + I;
      S3:=S3 + I * N[I];
      S4:=S4 + I * I;
  End;
  Bot1:=(S2 * S2) - (cnp * S4);
  Top1:=(S1 * S2) - (cnp * S3);
  afsd:=0;
  b:=0;
  If Bot1 <> 0 Then Begin
     B:=Top1/Bot1;
```

```
      afsd:=abs(s4/Bot1);
      afsd:=exp((0.5) * ln(afsd));
    End;
    a:=(S1 - B * s2)/ cnp;

End;   (* Procedure FindAandB *)
(*==========================================================================*)
Function Fnsd(pp:real):Real;

VAR
  X,Y,Z: real;

Begin  (* Function Fnsd *)

   X:=0;  Y:=0;
   X:=exp((2.479) * ln(pp));
   Y:=0.0000292 * X + 0.1/(100.0 - pp);
   Fnsd:=Y;

End;   (* Function Fnsd *)
(*==========================================================================*)

Begin

 GraphBackGround(Blue);
 TextBackGround(Blue);
 TextColor(Yellow);
 ClrScr;
 BoxWin(1,1,80,24,'  STEP 16 ');
 GotoXY(30,10);
 Write('>>> MIR Study <<<');
 GotoXY(1,1);
 GetDir(0,ProgDrvDir);
 Assign(GFN,'GFN_Nail');
 Reset(GFN);
 ChkDir:='';  ChkFile:=''; DrvDir:='';
 Readln(GFN,DrvDir);
 Readln(GFN,ChkFile);
 Close(GFN);
 ChkDir:=DrvDir;
 If Length(ChkDir) = 2 Then ChkDir:=ChkDir + '\';
 Delay(1000);
 IFileName:='';
 Default:=False;
 If ((ChkFile <> '**') and (Pos('DTB',ChkFile) > 0)) Then Begin
    IFileName:=ChkFile;
```

```
   Default:=True;
   If ProgDrvDir <> DrvDir Then AddDir(IFileName);
End
Else Begin
   Mark(HeapTop);
   GetDirectory('????????.dtb');
   ReqFile:=True;
    (* Assign Input File *)

   ValFile:=False;
   IFileName:='';
   While ValFile = False Do Begin
      IFileName:='                    ';
      FileName(IFileName,'  Input  ');
      AddFileExt(IFileName,'DTB');
      If ProgDrvDir <> DrvDir Then AddDir(IFileName);
      ValFile:=Exists(IFileName);
      If ValFile = False Then Begin
         GotoXY(1,12); ClrEOL;
         GotoXY(15,12); Write(^G,'File Does Not Exist !!');
         Delay(3000);
      End;
   End;
   Release(HeapTop);
End;
Assign(InFile,IFileName);
Reset(InFile);
ClrScr;
If Default = False Then Begin
   Repeat
     GotoXY(25,10);
     PrtOut:=' ';
     Write('Print Output [Y,N] ?: _');
     GotoXY(WhereX -1,WhereY);
     Read(Kbd,PrtOut);
     PrtOut:=UpCase(PrtOut);
     GotoXY(WhereX,WhereY); Write(PrtOut);
  Until ((PrtOut = 'Y') or (PrtOut = 'N'));
End
Else PrtOut:='N';
If PrtOut <> 'Y' Then Begin
   If ((ChkFile <> '**') and (Pos('DTB',ChkFile) > 0)) Then Begin
      OFileName:=ChkFile;
      AddFileExt(OFileName,'R16');
      If ProgDrvDir <> DrvDir Then AddDir(OFileName);
   End
   Else Begin
```

```
        Mark(HeapTop);
        GetDirectory('????????.r16');
        ValFile:=True;
        OFileName:='';
        While ValFile = True Do Begin
           ReqFile:=True;
           OFileName:='                    ';
           FileName(OFileName,'  Report  ');
           AddFileExt(OFileName,'R16');
           If ProgDrvDir <> DrvDir Then AddDir(OFileName);
           ValFile:=Exists(OFileName);
           If ValFile = True Then Begin
              GotoXY(1,12); ClrEOL;
              GotoXY(15,12); Write(^G,'File Already Exist !!');
              GotoXY(15,13); Write('Do you wish to overwrite?: __');
              GotoXY(WhereX -1,WhereY);
              Read(Kbd,Ans);
              Ans:=UpCase(Ans);
              GotoXY(WhereX,WhereY);
              Write(Ans);
              If Ans = 'Y' Then ValFile:=False;
           End;
        End;
        Release(HeapTop);
     End;
     Assign(PrtFile,OFileName);
     Rewrite(PrtFile);
  End;
  If Default = True Then ProjCon:=95
  Else Repeat
    ProjCon:=0;
    ClrScr;
    GotoXY(15,10);
    Write('Projection confidence level % (e.g. 95) : ___ %');
    HlpMsg1:='Enter the project confidence level.';
    HlpMsg2:='';
    Cl:=WhereX - 4;
    Rw:=WhereY;
    InputStr(ProjConStr,2,Cl,Rw,True);
    Val(ProjConStr,ProjCon,Code);
    If ProjCon > 100 Then Begin
       GotoXY(12,20);
       Write(^G,^G,'That will require an INFINITE number of repairs to ensure!!');
       Delay(3000);
    End;
  Until (ProjCon < 100.0);
  ClrScr;
```

```
NoMonths:=0;
NoYrs:=5;
SquProj:='N';
If Default = False Then Begin
   GotoXY(15,10);
   Write('Number of months past data to start projection: ___');
   HlpMsg1:='Enter the number of months past data to start projection.';
   HlpMsg2:='';
   Cl:=WhereX - 2; Rw:=WhereY;
   InputStr(NoMonthsStr,2,Cl,Rw,True);
   Val(NoMonthsStr,NoMonths,Code);
   GotoXY(15,13);
   Write('Number of years to do MIR projection for: ____');
   HlpMsg1:='Enter the number of years to do MIR projection.';
   HlpMsg2:='';
   Cl:=WhereX - 2;  Rw:=WhereY;
   InputStr(NoYrsStr,2,Cl,Rw,True);
   Val(NoYrsStr,NoYrs,Code);
   If NoYrs < 1 then NoYrs:=5;
   ClrScr;
   GotoXY(20,10);
   SquProj:=' ';
   Write('Is projection for squad [Y,N] ?: _');
   HlpMsg1:='Enter  Y  if projection is for a squadron, else enter  N .';
   HlpMsg2:='';
   Cl:=WhereX - 1; Rw:=WhereY;
   InputStr(ChkStr,1,Cl,Rw,False);
   If ChkStr <> ' ' Then SquProj:=ChkStr[1];
   SquProj:=UpCase(SquProj);
   If SquProj = 'Y' Then Begin
      I:=1;
      SquadCode:='';
      GotoXY(21,13);
      Write('Enter Squad. Code (e.g PX5): _____');
      HlpMsg1:='Enter the Squadron Code.';
      HlpMsg2:='';
      Cl:=WhereX -3; Rw:=WhereY;
      InputStr(SquadCode,3,Cl,Rw,False);
   End;
End;

ClrScr;
PrtDate:=Date;
PrtTime:=Time;
If PrtOut = 'Y' Then Begin
   Writeln(Lst,Chr(15));
   Writeln(Lst,'* Master Index of Repariables (MIR) *');
```

```
   Writeln(Lst);
   PrtComments('N');
   Writeln(Lst);
   Writeln(Lst,'  Data File used for study: ',IFileName);
   Writeln(Lst,'  Date of study:            ',PrtDate);
   Writeln(Lst,'  Time of study:            ',PrtTime);
   Writeln(Lst,'  Confidence level (%):     ',ProjCon:5,' %');
   If SquProj = 'Y' Then
      Writeln(Lst,'  Projection for ',SquadCode,' squad.');
   Writeln(lst,^L);
End
Else Begin
   Writeln(PrtFile,Chr(15));
   Writeln(PrtFile,'* Master Index of Repariables (MIR) *');
   Writeln(PrtFile);
   PrtComments('Y');
   Writeln(PrtFile);
   Writeln(PrtFile,'  Data File used for study: ',IFileName);
   Writeln(PrtFile,'  Date of study:            ',PrtDate);
   Writeln(PrtFile,'  Time of study:            ',PrtTime);
   Writeln(PrtFile,'  Confidence level (%):     ',ProjCon:5,' %');
   If SquProj = 'Y' Then
      Writeln(PrtFile,'  Projection for ',SquadCode,' squad.');
   Writeln(PrtFile,^L);
End;

GotoXY(20,10);
Write('Executing MIR Analysis and Projections');

t1:=99999.0; t2:=0;  dt:=0;
N1:=0;  n2:=0;  nobs:=1;
nlst:=nmax; ap:=0;  noi:=0;
noo:=0;  nod:=0;  nro:=0; np:=0;
For I:=1 to 600 Do n[I]:=0;

While (Not EOF(InFile)) Do Begin
   Readln(InFile,Line);
   wuc:=Copy(Line,9,7);
   bn:=Copy(Line,2,6);
   cn:=copy(Line,126,1);
   If cn = 'E' Then Begin
      Tsq:=Copy(Line,53,3);
      If ( ((SquProj = 'Y') and (SquadCode = Tsq)) or (SquProj <> 'Y'))
      Then Begin
         TStr:=Copy(Line,56,7); I:=1;
         Repeat
            If Tstr[I] = ' ' Then Delete(Tstr,I,1)
```

```
            Else I:=I + 1;
          Until (I >= Length(Tstr));
          Val(TStr,T,Code);
          ii:=Trunc(T/30.4) + 1;
          Tato:=Copy(Line,90,1);
          Tati:=Copy(Line,92,1);
          Tatd:=Copy(Line,94,1);
          If ((Tato = 'B') or (Tato = 'C') or (Tato = 'R')) then noo:=noo + 1;
          If Tato = 'R' Then nro:=nro + 1;
          If ((Tati = 'B') or (Tati = 'C') or (Tati = 'R')) then noi:=noi + 1;
          If ((Tatd = 'B') or (Tatd = 'C') or (Tatd = 'R')) then nod:=nod + 1;
          If T > T2 Then T2:=T;
          If ii > nmax then ii:=nmax;
          If ((ii > nobs) and (ii < nmax)) then nobs:=ii;
          If ((ii < nlst) and (ii > 0)) then nlst:=ii;
          n[ii]:=n[ii] + 1;
          n1:=n1 + 1;
          If T < T1 Then T1:=T;
          FStr:=Copy(Line,129,6); I:=1;
          Repeat
            If Fstr[I] = ' ' Then Delete(Fstr,I,1)
            Else I:=I + 1;
          Until (I >= Length(Fstr));
          Val(FStr,F,Code);
          If F > 0 Then Dt:=Dt + (T - F);
          npStr:=Copy(Line,135,3); I:=1;
          Repeat
            If Npstr[I] = ' ' Then Delete(Npstr,I,1)
            Else I:=I + 1;
          Until (I >= Length(Npstr));
          Val(NpStr,np,Code);
          If Np = 0 Then ap:=ap + 1;
        End;
    End;
End;
If n1 > 0 Then Begin
  n2:=ap;
  nodg:=nro - noi;
  fo:=noo/n1; fi:=noi/n1;  fd:=nod/n1;
  If fd = 0 Then Begin
     ndod:=1;
     fd:=nodg/n1;
  end;
  FindAandB;
  sd2:=0;
  advc:=0; advn:=0;
  clrScr;
```

```
Writeln('   Number of Repairs/Removals    ');
Writeln;
Writeln('  Date  ','Obs.':10,'fit':7,'del.':7);
Writeln('--------','----':10,'----':7,'----':7);
If PrtOut = 'Y' Then Begin
   Writeln(Lst,'   Number of Repairs/Removals    ');
   Writeln(Lst);
   Writeln(Lst,'  Date  ','Obs.':10,'fit':7,'del.':7);
   Writeln(Lst,'--------','----':10,'----':7,'----':7);
End
Else Begin
   Writeln(PrtFile,'   Number of Repairs/Removals    ');
   Writeln(PrtFile);
   Writeln(PrtFile,'  Date  ','Obs.':10,'fit':7,'del.':7);
   Writeln(PrtFile,'--------','----':10,'----':7,'----':7);
End;

SFileName:='SqrDqt.sqr';
If DrvDir <> ProgDrvDir Then AddDir(SFileName);
Assign(SFile,SFileName);
ReSet(SFile);
Seek(SFile,0);
Read(SFile,SqdRec);
Close(SFile);
SqdLine:=SqdRec.SLine;
ByStr:=Copy(SqdLine,7,2);
Val(ByStr,BY,Code);

For I:= nlst to nobs do Begin
    nc:=A + B * I;
    dn:=n[I] - nc;
    sd2:=sd2 + dn * dn;
    advc:=advc + nc;
    advn:=advn + n[I];

    If I MOD 12 <> 0 Then Begin
       PrtYr:=BY + (I DIV 12);
       ChkMonth:=I MOD 12;
    End
    Else Begin
       PrtYr:=BY + ( (I DIV 12) - 1);
       ChkMonth:=12;
    End;
```

```
    Case ChkMonth of
        1: PrtMonth:='JAN. ';
        2: PrtMonth:='FEB. ';
        3: PrtMonth:='MAR. ';
        4: PrtMonth:='APR. ';
        5: PrtMonth:='MAY  ';
        6: PrtMonth:='JUNE ';
        7: PrtMonth:='JUL. ';
        8: PrtMonth:='AUG. ';
        9: PrtMonth:='SEPT.';
       10: PrtMonth:='OCT. ';
       11: PrtMonth:='NOV. ';
       12: PrtMonth:='DEC. ';
    End;
    writeln(PrtMonth,PrtYr:3,n[I]:10,nc:7:0,dn:7:0);
    If PrtOut = 'Y' then
       writeln(Lst,PrtMonth,PrtYr:3,n[I]:10,nc:7:0,dn:7:0)
    Else
       writeln(PrtFile,PrtMonth,PrtYr:3,n[I]:10,nc:7:0,dn:7:0)
end;
advn:=advn/cnp;
advc:=advc/cnp;
If cnp > 1 Then Begin
  sd:=sd2/(cnp - 1);
  sd:=exp((0.5) * ln(sd));
End
Else sd:=0;
Writeln('====':18,'====':7,'====':7);
Writeln('Adv.',advc:14:0,advc:7:0,sd:7:0);
If PrtOut = 'Y' then Begin
   Writeln(Lst,'====':18,'====':7,'====':7);
   Writeln(Lst,'Adv.',advc:14:0,advc:7:0,sd:7:0);
   Writeln(lst,^L);
End
Else Begin
   Writeln(PrtFile,'====':18,'====':7,'====':7);
   Writeln(PrtFile,'Adv.',advc:14:0,advc:7:0,sd:7:0);
   Writeln(PrtFile,^L);
End;
Mtbf:=0; DDT:=0; Ta:=0; VMax:=0;

mtbf:=(T2 - T1) * n2/n1;
DDT:=(T2 - T1)/nobs;
ta:=mtbf - dt/n1;
If n2 > 0 then ta:=ta * n1 / n2;
For K:=1 to 2 Do Begin
   If K = 1 then I:=Trunc(t2/30.4) + 1
```

```
    Else I:=Trunc(t1/30.4) + 1;
    If I MOD 12 <> 0 Then Begin
       PrtYr:=BY + (I DIV 12);
       ChkMonth:=I MOD 12;
    End
    Else Begin
       PrtYr:=BY + ( (I DIV 12) - 1);
       ChkMonth:=12;
    End;

    Case ChkMonth of
         1: PrtMonth:='JAN. ';
         2: PrtMonth:='FEB. ';
         3: PrtMonth:='MAR. ';
         4: PrtMonth:='APR. ';
         5: PrtMonth:='MAY  ';
         6: PrtMonth:='JUNE ';
         7: PrtMonth:='JUL. ';
         8: PrtMonth:='AUG. ';
         9: PrtMonth:='SEPT.';
        10: PrtMonth:='OCT. ';
        11: PrtMonth:='NOV. ';
        12: PrtMonth:='DEC. ';
    End;
    If K = 1 Then Begin
       PrtM1:=PrtMonth;
       PrtY1:=PrtYr;
    End
    Else Begin
       PrtM2:=PrtMonth;
       PrtY2:=PrtYr;
    End;
End;
vMax:=1.5 * ta;
Writeln('Max Time in data        =',t2:6:0,'   ',PrtM1,PrtY1:3);
Writeln('Min time in data        =',t1:6:0,'   ',PrtM2,PrtY2:3);
Writeln('Total number of data    =',n1:6);
Writeln('Number to be cal.       =',n2:6);
Writeln('Mean time between fail. =',mtbf:10:3);
Writeln('Mean value for unseen   =',ta:10:3);
Writeln('Max time for cal. time  =',vmax:10:3);
Writeln('-----------------------------------');
Writeln('Number of repairs/removals (O-level) =',noo:4);
Writeln('Number of repairs/removals (I-level) =',noi:4);
Writeln('Assumed   repairs/removals (D-level) =',nodg:4);
Writeln('Observed  repairs/removals (D-level) =',nod:4);
```

```
If prtOut = 'Y' Then Begin
   Writeln(Lst,'Max Time in data        =',t2:6:0,'   ',PrtM1,PrtY1:3);
   Writeln(Lst,'Min time in data        =',t1:6:0,'   ',PrtM2,PrtY2:3);
   Writeln(Lst,'Total number of data    =',n1:6);
   Writeln(Lst,'Number to be cal.       =',n2:6);
   Writeln(Lst,'Mean time between fail. =',mtbf:10:3);
   Writeln(Lst,'Mean value for unseen   =',ta:10:3);
   Writeln(Lst,'Max time for cal. time  =',vmax:10:3);
   Writeln(Lst,'-----------------------------------');
   Writeln(Lst,'Number of repairs/removals (O-level) =',noo:4);
   Writeln(Lst,'Number of repairs/removals (I-level) =',noi:4);
   Writeln(Lst,'Assumed   repairs/removals (D-level) =',nodg:4);
   Writeln(Lst,'Observed  repairs/removals (D-level) =',nod:4);
End
Else Begin
   Writeln(PrtFile,'Max Time in data        =',t2:6:0,'   ',PrtM1,PrtY1:3);
   Writeln(PrtFile,'Min time in data        =',t1:6:0,'   ',PrtM2,PrtY2:3);
   Writeln(PrtFile,'Total number of data    =',n1:6);
   Writeln(PrtFile,'Number to be cal.       =',n2:6);
   Writeln(PrtFile,'Mean time between fail. =',mtbf:10:3);
   Writeln(PrtFile,'Mean value for unseen   =',ta:10:3);
   Writeln(PrtFile,'Max time for cal. time  =',vmax:10:3);
   Writeln(PrtFile,'-----------------------------------');
   Writeln(PrtFile,'Number of repairs/removals (O-level) =',noo:4);
   Writeln(PrtFile,'Number of repairs/removals (I-level) =',noi:4);
   Writeln(PrtFile,'Assumed   repairs/removals (D-level) =',nodg:4);
   Writeln(PrtFile,'Observed  repairs/removals (D-level) =',nod:4);
End;
If Ndod <> 0 then Begin
   Writeln('* No data on D - level so best aprox. made. *');
   If PrtOut = 'Y' Then
      Writeln(Lst,'* No data on D - level so best aprox. made. *')
   Else
      Writeln(PrtFile,'* No data on D - level so best aprox. made. *')
End;
If PrtOut = 'Y' Then Writeln(lst,^L)
Else Write(PrtFile,^L);

ndl:=Fnsd(ProjCon);
SqrRootTwelve:=Exp((0.5) * ln(12.0));
sdx:=ndl * sd * SqrRootTwelve;
afsd:=afsd * ndl * SqrRootTwelve;
nobs:=nobs + NoMonths;
Proj:=nobs + 60;
If Proj > NMax Then Proj:=NMax;
icy:=0;  icm:=0;  sno:=0;  sni:=0;  snd:=0;
```

```
  For I:=nobs to proj Do Begin
      nc:= A + B * I;
      nct:=nc;
      nco:=nct * fo;
      nci:=nct * fi;
      ncd:=nct * fd;
      sno:=sno + nco;
      sni:=sni + nci;
      snd:=snd + ncd;
      icm:=icm + 1;
      If icm >= 12 Then Begin
         icm:=0; icy:=icy + 1;
         prjo[icy]:=sno + sdx * fo;
         prji[icy]:=sni + sdx * fi;
         prjd[icy]:=snd + sdx * fd;
         sxd:=sdx + afsd;
         sno:=0; sni:=0;snd:=0;
     end;
  End;
  For I:=1 to icy do begin
     tt:=prjo[I] - prji[I];
     If prjd[I] > tt Then prjd[I]:=tt;
     prjo[I]:=int(prjo[i]);
     prji[I]:=int(prji[I]);
     prjd[I]:=int(prjd[I]);
  End;
End
Else Begin
 ClrScr;
  If SquProj = 'Y' Then Begin
     GotoXY(18,10);
     Writeln(^G,'No Records Found With Squad.Code  ',SquadCode);
  End
  Else Begin
     GotoXY(32,10);
     Writeln(^G,'EMPTY FILE');
  End;
End;
Close(InFile);

If n1 > 0 Then Begin
   I:=1;
   Writeln('* Master Index of Repariables (MIR) *');
   If SquProj = 'Y' Then
      Writeln(' [ For squad.  ',SquadCode,' ]');
   Writeln;
   Writeln('WUC: ',Wuc);
```

```
Writeln;
Writeln;
Writeln('Year        Operational        Intermediate          Depot');
Writeln('----        -----------         ------------          -----');
Writeln;
If PrtOut = 'Y' Then Begin
   Writeln(Lst,'* Master Index of Repariables (MIR) *');
   If SquProj = 'Y' Then
      Writeln(Lst,' [ For squad.  ',SquadCode,' ]');
   Writeln(lst);
   Writeln(Lst,'WUC: ',Wuc);
   Writeln(Lst);
   Writeln(Lst);
   Writeln(Lst,'Year        Operational        Intermediate          Depot');
   Writeln(Lst,'----        -----------         ------------          -----');
   Writeln(Lst);
End
Else Begin
   Writeln(PrtFile,'* Master Index of Repariables (MIR) *');
   If SquProj = 'Y' Then
      Writeln(PrtFile,' [ For squad.  ',SquadCode,' ]');
   Writeln(PrtFile);
   Writeln(PrtFile,'WUC: ',Wuc);
   Writeln(PrtFile);
   Writeln(PrtFile);
   Writeln(PrtFile,'Year        Operational        Intermediate          Depot');
   Writeln(PrtFile,'----        -----------         ------------          -----');
   Writeln(PrtFile);
End;
TotPrjo:=0;
TotPrji:=0;
TotPrjd:=0;
For J:=1 to icy Do Begin
    Writeln(J:3,Prjo[J]:15:0,Prji[J]:23:0,Prjd[J]:21:0);
    TotPrjo:=TotPrjo + Prjo[J];
    TotPrji:=TotPrji + Prji[J];
    TotPrjd:=TotPrjd + Prjd[J];
    If PrtOut = 'Y' Then
       Writeln(Lst,J:3,Prjo[J]:15:0,Prji[J]:23:0,Prjd[J]:21:0)
    Else
       Writeln(PrtFile,J:3,Prjo[J]:15:0,Prji[J]:23:0,Prjd[J]:21:0);

End;
Writeln('==================================================================');
Writeln('Total',TotPrjo:13:0,TotPrji:23:0,TotPrjd:21:0);
Writeln('Adv.',TotPrjo/icy:14:0,TotPrji/icy:23:0,TotPrjd/icy:21:0);
If PrtOut = 'Y' Then Begin
```

```
        Writeln(Lst,'==================================================================');
        Writeln(Lst,'Total',TotPrjo:13:0,TotPrji:23:0,TotPrjd:21:0);
        Writeln(Lst,'Adv.',TotPrjo/icy:14:0,TotPrji/icy:23:0,TotPrjd/icy:21:0);
    End
    Else Begin
        Writeln(PrtFile,'==================================================================');
        Writeln(PrtFile,'Total',TotPrjo:13:0,TotPrji:23:0,TotPrjd:21:0);
        Writeln(PrtFile,'Adv.',TotPrjo/icy:14:0,TotPrji/icy:23:0,TotPrjd/icy:21:0);
    End;

 End;
 GotoXY(1,24);
 Write('Press any key to Continue');
 Read(kbd,ch);
 If PrtOut <> 'Y' Then Begin
    Close(PrtFile);
    TempCom:='MIR REPORT CREATED USING INPUT FILE ' + IFileName;
    UpdateCom;
 End;

End.^Z
```

[BLANK PAGE]

Appendix J.25

MonthFail

```
Program MonthFail(input,output);
{$C-}
```

STEP 8 - Monthly failure rate. This step does an initial evaluation of the number of failures per month for a given time interval. Theh mean time between failures (MTBF) and other values are estimated. The monthly failure rate is fit to a Weibull projection equation.

Deborah A. Holder

```
LABEL
   TRYAGAIN;

TYPE
   Array70 = Array [1..70] of char;
   String255 = string[255];
   String70 = String[70];
   String50 = String[50];
   String20 = String[20];
   String10 = String[10];
   String3  = String[3];
   String150 = string[150];
   Char12arr = array [1..12] of char;
   HoldName = array [1..25] of char;
   RegRec = record
              ax,bx,cx,dx,bp,si,di,ds,es,flags: integer;
            end;

   DPointer = ^DirRec;
   DirRec = record
              Size: real;
              Hr,Minu: integer;
              Year,Month,Day: integer;
              Next,Prev: DPointer;
              Name: String20;
            End;

   SqrRec = record
               SqrLine:Array70;
              end;
```

```
  SqrString = string[10];
  Path = String[64];
  ImageType = array [1..4096] of char;
  DateTimeType = String[8];

VAR
  Sqrdqt: File of SqrRec;
  OFileName,CFilename,SFileName,IFileName: Path;
  GFN,INFile,PrtFile,ChangeFile: text;
  PrtOpt: char;
  HeapTop: ^Integer;
  NoFiles: integer;
  Start,TheEnd: Integer;
  Top,Bottom: DPointer;
  FirstRec,LastRec,NewRec: DPointer;
  FileEnd,I,J,K,L: integer;
  TQ,Ts,ii,Code: integer;
  Csq,Sq: integer;
  Fsq,IntSQ: array [1..50,1..50] of real;
  Fdt,Av,Tfs,Xsq,Nsq: array [1..50] of real;
  SqrIn: SqrRec;
  Isb,VN: real;
  BegMonth: integer;
  PrtY1,PrtY2,PrtYr,ChkMonth: integer;
  PrtM1,PrtM2,PrtMonth: String[5];
  MonthNo,N,NCO: array [1..50] of integer;
  ChkVal,HoldVal,HoldYr:integer;
  UppCase,First: Boolean;
  F,T,T1,T2,DT,AR,N1,N2: real;
  Np,Bot,Mtbf,DDT,TT,Ta,Max: real;
  Tv,Cdt,Dc,Dv,Se,Sev,Nvd,Vdt,Ncd,V,V2,Nt,Ns,Nv,Nv2,A,B: real;
  Ix,I1,I48: integer;
  AA,BB: array [1..10] of real;
  ChkLine: String70;
  SqStrArr: array [1..50] of SqrString;
  ChkSqd: SqrString;
  CpStr: SqrString;
  SqrFileName,CurDir: Path;
  Ans,NewDrive: char;
  Default,Quit,Found,ReqFile,ValFile: Boolean;
  Replace,Initial: integer;
  Field,MaxField: Integer;
  Line: String150;
  NpStr,BN,TStr,FStr:String10;
  PrtDate,PrtTime: DateTimeType;
  Image: ImageType;
  CrtMode: Byte absolute $0040:$0049;
```

```
    MonoBuffer: ImageType absolute $B000:$0000;
    ColorBuffer: ImageType absolute $B800:$0000;
    ChkFile,ChkDir,ProgDrvDir,DrvDir: Path;
    TempCom: String255;
    TmpCom1,TmpCom2,TmpCom3,TmpCom4,TmpCom5: Path;
    RdStr: path;
    PrtTitle: String70;
    HlpMsg1,HlpMsg2: String255;

{$IBoxWin.pas}
{$IHelp.pas}
{$IAddDir.pas}
{$IComments.pas}
{$IExists.pas}
{$IDiskSpace.pas}
{$IGetDir.pas}
{$IDirlst.pas}
{$IFileName.pas}
{$IAddFileExt.pas}
{$ITime.pas}
{$IDate.pas}
{$IPrtComments.pas}
{$IReadField.pas}
{$IReadCom.pas}
{$IDisCom.pas}
{$IUpdateCom.pas}

(*==========================================================================*)
  Procedure PrintValPrt;

  VAR
     K,L: integer;

   Begin (* Procedure PrintValPrt *)

      If Sq > 24 Then K:=2
      Else K:=1;
      For L:=1 to K Do Begin
          Write(Lst,PrtTitle);
          Writeln(Lst,'Sheet ':63,L:2,' of ',K:2);
          For I:=1 to 132 Do Write(Lst,'=');
          Writeln(Lst);
          Writeln(Lst,'  Date    |-------- Sqr. Number -------->');
          Write(Lst,'       ');
          If Sq > 24 Then Begin
             If L = 1 Then Begin
                Start:=1;  TheEnd:=24;
```

```
   End
   Else Begin
      Start:=25;  TheEnd:=sq;
   End;
End
Else Begin
   Start:=1;  TheEnd:=sq;
End;
For I:=Start to TheEnd Do Begin
  Tfs[I]:=0;
  Write(Lst,I:5);
End;
Writeln(Lst,'    Adv.'); TQ:=0;
For I:=1 to 48 Do Begin
   If MonthNo[I] MOD 12 <> 0 Then Begin
      PrtYr:=HoldYr + (MonthNo[I] DIV 12);
      ChkMonth:=MonthNo[I] MOD 12;
   End
   Else Begin
      PrtYr:=HoldYr + ((MonthNo[I] DIV 12) -1);
      ChkMonth:=12;
   End;
   Case ChkMonth of
      1: PrtMonth:='JAN. ';
      2: PrtMonth:='FEB. ';
      3: PrtMonth:='MAR. ';
      4: PrtMonth:='APR. ';
      5: PrtMonth:='MAY  ';
      6: PrtMonth:='JUNE ';
      7: PrtMonth:='JULY ';
      8: PrtMonth:='AUG. ';
      9: PrtMonth:='SEP. ';
     10: PrtMonth:='OCT. ';
     11: PrtMonth:='NOV. ';
     12: PrtMonth:='DEC. ';
   End;

   Write(Lst,PrtMonth,PrtYr:2);
   For J:=Start to TheEnd Do Begin
      Tfs[J]:=Tfs[J] + Fsq[I,J];
      If Fsq[I,J] <> 0 Then
         Write(Lst,Fsq[I,J]:5:1)
      Else Write(Lst,'  ---');
      If Fsq[I,J] > 0 Then
         TQ:=TQ + 1;
   End;
   Writeln(Lst,Xsq[I]:6:1); TQ:=0;
```

```
        End;
        For I:=1 to 132 Do Write(Lst,'-');
        Writeln(Lst);
        Write(Lst,'Tot.   ');
        For I:=Start to TheEnd Do
           Write(Lst,Tfs[I]:5:1);
        Writeln(Lst);
        Write(Lst,'Adv.   ');
        For I:=Start to TheEnd Do Begin
            Bot:=Av[I];  If Bot = 0 then Bot:=1;
            Write(Lst,(Tfs[I]/Bot):5:1);
        End;
        Writeln(Lst);
        Writeln(Lst,^L);
     End;

  End; (* Procedure PrintValPrt *)
(*=========================================================================*)
  Procedure PrintValFile;

  VAR
     K,L: integer;

  Begin (* Procedure PrintValFile *)

     If Sq > 24 Then K:=2
     Else K:=1;
     For L:=1 to K Do Begin
        Write(PrtFile,PrtTitle);
        Writeln(PrtFile,'Sheet ':63,L:2,' of ',K:2);
        For I:=1 to 132 Do Write(PrtFile,'=');
        Writeln(PrtFile);
        Writeln(PrtFile,'  Date    |-------- Sqr. Number -------->');
        Write(PrtFile,'       ');
        If Sq > 24 Then Begin
           If L = 1 Then Begin
              Start:=1;  TheEnd:=24;
           End
           Else Begin
              Start:=25;  TheEnd:=sq;
           End;
        End
        Else Begin
           Start:=1;  TheEnd:=sq;
        End;
        For I:=Start to TheEnd Do Begin
```

```
    Tfs[I]:=0;
    Write(PrtFile,I:5);
End;
If ((( Sq > 24) and (L = 2)) or (Sq <=24)) Then
   Writeln(PrtFile,'    Adv.')
Else Writeln(PrtFile);
TQ:=0;
For I:=1 to 48 Do Begin
    If MonthNo[I] MOD 12 <> 0 Then Begin
       PrtYr:=HoldYr + (MonthNo[I] DIV 12);
       ChkMonth:=MonthNo[I] MOD 12;
    End
    Else Begin
       PrtYr:=HoldYr + ((MonthNo[I] DIV 12) -1);
       ChkMonth:=12;
    End;
    Case ChkMonth of
       1: PrtMonth:='JAN. ';
       2: PrtMonth:='FEB. ';
       3: PrtMonth:='MAR. ';
       4: PrtMonth:='APR. ';
       5: PrtMonth:='MAY  ';
       6: PrtMonth:='JUNE ';
       7: PrtMonth:='JULY ';
       8: PrtMonth:='AUG. ';
       9: PrtMonth:='SEP. ';
      10: PrtMonth:='OCT. ';
      11: PrtMonth:='NOV. ';
      12: PrtMonth:='DEC. ';
    End;

    Write(PrtFile,PrtMonth,PrtYr:2);
    For J:=Start to TheEnd Do Begin
       Tfs[J]:=Tfs[J] + Fsq[I,J];
       If Fsq[I,J] <> 0 Then
          Write(PrtFile,Fsq[I,J]:5:1)
       Else Write(PrtFile,'  ---');
       If Fsq[I,J] > 0 Then
          TQ:=TQ + 1;
    End;
    If ((( Sq > 24) and (L = 2)) or (Sq <=24)) Then
        Writeln(PrtFile,Xsq[I]:6:1)
    Else Writeln(PrtFile);
    TQ:=0;
End;
For I:=1 to 132 Do Write(PrtFile,'-');
Writeln(PrtFile);
```

```
        Write(PrtFile,'Tot.   ');
        For I:=Start to TheEnd Do
           Write(PrtFile,Tfs[I]:5:1);
        Writeln(PrtFile);
        Write(PrtFile,'Adv.   ');
        For I:=Start to TheEnd Do Begin
            Bot:=Av[I];  If Bot = 0 then Bot:=1;
            Write(PrtFile,(Tfs[I]/Bot):5:1);
        End;
        Writeln(PrtFile);
        Writeln(PrtFile,^L);
     End;

  End; (* Procedure PrintValFile *)
(*=================================================================*)
  Procedure WibelFit;

  VAR
    S1,S2,S3,S4: real;
    S1S,S2S,S3S,S4S: string20;
    N48,V,T1,F1,Top,Bot: real;
    Code,I: integer;

  Begin  (* Procedure WibelFit *)

     N48:=0;  S1:=0;  S2:=0;  S3:=0;  S4:=0;
     For I:=I1 to I48  Do Begin
         V:=0;  N48:=N48 + 1;
         If N[I] > 0 Then V:=Nsq[I]/N[I];
         If V = 0 Then V:=1;
         V:=DDT * V;  Fdt[I]:=V;
         T1:=ln(I * 30.4);
         F1:=ln(V);
         S1:=S1 + F1;
         S2:=S2 + T1;
         S3:=S3 + F1*T1;
         S4:=S4 + T1*T1;
     End;
     Top:=S1*S2 - N48*S3;
     Bot:=S2*S2 - N48*S4;
     B:=0;  If Bot <> 0 Then B:=Top/Bot;
     A:=(S1 - B*S2)/N48;
     A:=Exp(A);
     If PrtOpt = 'Y' Then Begin
        Writeln(Lst,' Time period from ',I1,' to ',I48);
        Writeln(Lst,' log(tbf) versus log(t) fit gives ');
        Writeln(Lst,' tbf = a*t**b');
```

```
        Writeln(Lst,' a   = ',A);
        Writeln(Lst,' b   = ',B);
        For I:=1 to 20 Do Write(Lst,'=');
        Writeln(Lst);
     End
     Else Begin
        Writeln(PrtFile,' Time period from ',I1,' to ',I48);
        Writeln(PrtFile,' log(tbf) versus log(t) fit gives ');
        Writeln(PrtFile,' tbf = a*t**b');
        Writeln(PrtFile,' a   = ',A);
        Writeln(PrtFile,' b   = ',B);
        For I:=1 to 20 Do Write(PrtFile,'=');
        Writeln(PrtFile);
     End;

  End;  (* Procedure WibelFit *)
(*=========================================================================*)
Begin  (* Program MonthFail *)

   GraphBackGround(Blue);
   TextBackGround(Blue);
   TextColor(Yellow);
   ClrScr; Replace:=0;
   BoxWin(1,1,80,24,'  STEP 8  ');
   GotoXY(25,10);
   Write('>>> Monthly Failure Rate <<<');
   GotoXY(1,1);
   SqrFileName:='SqrDqt.sqr';
   Default:=False;
   GetDir(0,ProgDrvDir);
   ChkDir:=''; ChkFile:=''; DrvDir:='';
   Assign(GFN,'GFN_Nail');
   Reset(GFN);
   Readln(GFN,DrvDir);
   Readln(GFN,ChkFile);
   If ChkFile <> '**' Then Default:=True;
   Close(GFN);
   ChkDir:=DrvDir;
   If Length(ChkDir) = 2 Then ChkDir:=ChkDir + '\';
   Delay(1000);
   ClrScr;
   If Default = False Then Begin
      GotoXY(10,5);
      Write('This program requires an Input file named Sqrdqt.sqr.  This file');
      GotoXY(10,6);
      Write('is assumed to reside on the ',DrvDir[1],' Drive.  If this drive does not');
      GotoXY(10,7);
```

```
      Write('contain the required file, input the correct drive: __');
      GotoXY(WhereX -1, WhereY);
      Readln(trm,NewDrive);
      NewDrive:=Upcase(NewDrive);
      If Ord(NewDrive) <> 13 Then
         SqrFileName:=NewDrive + ':' + SqrFileName
      Else If ProgDrvDir <> DrvDir Then AddDir(SqrFileName);
   End
   Else If ProgDrvDir <> DrvDir Then AddDir(SqrFileName);
   SFileName:='';
   SFileName:=SqrFileName;
   ValFile:=Exists(SFileName);
   If ValFile = True Then Begin
      Assign(Sqrdqt,SFileName);
      Reset(Sqrdqt);
      FileEnd:=FileSize(Sqrdqt) - 1;
      UppCase:=True;
      If Default = False Then Begin
         ClrScr;
         GotoXY(11,10);
         Write('Do you want to do change over calculations [Y/N]?: __');
         HlpMsg1:='Enter  Y  if you wish to change over calculations';
         HlpMsg2:='';
         RdStr:='';
         MaxField:=1; Field:=1; Quit:=False;
         ReadField(RdStr,WhereX -1,WhereY,1);
         If RdStr <> '' Then Ans:=UpCase(RdStr[1])
         Else Ans:='N';
         CFileName:='';
         If Ans = 'Y' Then Begin
            Mark(HeapTop);
            GetDirectory('????????.???');
TRYAGAIN:
            CFileName:='                    ';
            ReqFile:=False;
            FileName(CFileName,'ChangeOver');
            If CFileName <> '                    ' Then Begin
               If ProgDrvDir <> DrvDir Then AddDir(CFileName);
               ValFile:=Exists(CFileName);
               If ValFile = True Then Begin
                  Assign(ChangeFile,CFileName);
                  Reset(ChangeFile);
                  For I:=1 to 50 do NCO[I]:=0;
                  I:=1;
                  While ((Not EOF(ChangeFile)) and (I <= 48)) Do Begin
                        Readln(ChangeFile,J,NCO[I]);
                        I:=I + 1;
```

```
            End;
            Close(ChangeFile);
         End
         Else Begin
            ClrScr;
            GotoXY(15,12);
            Write(^G,'File does not exist');
            Delay(3000);
            Goto TRYAGAIN;
        End;
      End; (* CFileName = Blanks *)
      Release(HeapTop);
      ClrScr;
      Initial:=0;
      GotoXY(11,10);
      Write('Do you want to replace with change over inv. [Y/N]?: __');
      HlpMsg1:='Enter  Y  if you wish to replace with change over inv.';
      HlpMsg2:='';
      RdStr:='';
      MaxField:=1; Field:=1; Quit:=False;
      ReadField(RdStr,WhereX -1,WhereY,1);
      If RdStr <> '' Then Ans:=UpCase(RdStr[1])
      Else Ans:='N';
      Replace:=-1;
      If Ans = 'Y' Then Begin
         GotoXY(11,12);
         Write('Do you want to add initial aircraft to inv. [Y/N]?: __');
         HlpMsg1:='Enter  Y  if you wish to add initial aircraft to inv.';
         HlpMsg2:='';
         RdStr:='';
         MaxField:=1; Field:=1; Quit:=False;
         ReadField(RdStr,WhereX -1,WhereY,1);
         If RdStr <> '' Then Ans:=UpCase(RdStr[1])
         Else Ans:='N';
         If Ans = 'Y' Then Begin
            GotoXY(11,14);
            Write('Number to be added to change over as inv. = ');
            HlpMsg1:='Enter number to be added to change over inv.';
            HlpMsg2:='';
            RdStr:='';
            MaxField:=1; Field:=1; Quit:=False;
            ReadField(RdStr,WhereX,WhereY,5);
            Val(RdStr,Initial,Code);
         End;
         Replace:=1;
      End;
   End;  (* Ans = Y *)
End;
```

```
IFileName:='';
If ((ChkFile <> '**') and (Pos('DTB',ChkFile) > 0)) Then Begin
   IFileName:=ChkFile;
   If DrvDir <> ProgDrvDir Then AddDir(IFileName);
End
Else Begin
   Mark(HeapTop);
   GetDirectory('????????.DTB');
   ReqFile:=True;
   ValFile:=False;
   IFileName:='';
   While ValFile = False Do Begin
       ReqFile:=True;
       IFileName:='                    ';
       Filename(IFileName,' Input   ');
       AddFileExt(IFileName,'DTB');
       If DrvDir <> ProgDrvDir Then AddDir(IFileName);
       ValFile:=Exists(IFileName);
       If ValFile = False Then Begin
          GotoXY(1,12); ClrEOL;
          GotoXY(15,12); Write(^G,'File Does not Exist !!');
          Delay(3000);
       End;
   End;
   Release(HeapTop);
End;

Assign(InFile,IFileName);
Reset(InFile);

ClrScr;
If Default = False Then Begin
   GotoXY(10,10);
   Write('Do you wish to send output directly to printer?: __');
   GotoXY(WhereX - 1,WhereY);
   Read(Kbd,PrtOpt);
   PrtOpt:=UpCase(PrtOpt);
   GotoXY(WhereX, WhereY);
   Write(PrtOpt);
End
Else PrtOpt:= 'N';
If PrtOpt <> 'Y' Then Begin
  If ((ChkFile <> '**') and (Pos('DTB',ChkFile) > 0)) Then Begin
     OFileName:=ChkFile;
     AddFileExt(OFileName,'R08');
     If DrvDir <> ProgDrvDir Then AddDir(OFileName);
  End
```

```
  Else Begin
   Mark(HeapTop);
   GetDirectory('????????.r08');
   ValFile:=True;
   OFileName:='';
   While ValFile = True Do Begin
       ReqFile:=True;
       OFileName:='                    ';
       Filename(OFileName,' Report ');
       AddFileExt(OFileName,'R08');
       If DrvDir <> ProgDrvDir Then AddDir(OFileName);
       ValFile:=Exists(OFileName);
       If ValFile = True Then Begin
          GotoXY(1,12); ClrEOL;
          GotoXY(15,12); Write(^G,'File Already Exist !!');
          GotoXY(15,13); Write('Do you wish to overwrite?: _');
          GotoXY(WhereX -1,WhereY);
          Read(Kbd,Ans);
          Ans:=UpCase(Ans);
          GotoXY(WhereX,WhereY);
          Write(Ans);
          If Ans = 'Y' Then ValFile:=False;
       End;
   End;
   Release(HeapTop);
  End;
  Assign(PrtFile,OFileName);
  Rewrite(PrtFile);
End;
ClrScr;
GotoXY(23,10);
Write('CALCULATING MONTHLY FAILURE RATE');
Csq:=0; Sq:=0;
For I:=1 to 50 Do Begin
    SqStrArr[I]:='';
    For J:=1 to 50 Do Begin
        IntSq[I,J]:=0; Av[I]:=0;

    End;
    Nsq[I]:=0;
    N[I]:=0;
End;
Seek(Sqrdqt,0);
Read(Sqrdqt,Sqrin);
ChkLine:='';
ChkLine:=Sqrin.SqrLine;
CpStr:=Copy(ChkLine,7,2);
```

```
Val(CpStr,HoldYr,Code);
CpStr:=Copy(ChkLine,1,4);
If CpStr[1] = ' ' Then Delete(CpStr,1,1);
Val(CpStr,HoldVal,Code);
Delay(3000);
For I:=1 to FileEnd Do Begin
   Seek(Sqrdqt,I);
   Read(Sqrdqt,SqrIn);
   ChkLine:='';
   ChkLine:=SqrIn.SqrLine;
   If Pos('ORG :',ChkLine) = 0 Then Begin
      If Csq <> 0 Then Begin
        CpStr:=Copy(ChkLine,6,5);
        J:=1;
        Repeat
          If CpStr[J] = ' ' Then
              Delete(CpStr,J,1)
          Else J:=J + 1;
        Until (J >= Length(CpStr));
        Val(CpStr,VN,Code);
        CpStr:=Copy(ChkLine,1,4);
        If CpStr[1] = ' ' Then Delete(CpStr,1,1);
        Val(CpStr,ChkVal,Code);
        If ChkVal = HoldVal Then Csq:=1
        Else Csq:=Trunc((ChkVal - HoldVal)/30.4 + 0.5) + 1;
        IntSq[Csq,Sq]:=VN;
        Csq:=Csq + 1;
      End;
   End
   Else Begin
      CpStr:=Copy(ChkLine,6,3);
      Csq:=1; Sq:=Sq + 1;
      SqStrArr[Sq]:=CpStr;
   End;
End;
BegMonth:=Trunc(HoldVal/30.4);
MonthNo[1]:=BegMonth;
For I:=2 to 48 Do
    MonthNo[I]:=MonthNo[I - 1] + 1;
For I:=Sq + 1 to 50 Do SqStrArr[I]:='   ';
PrtDate:=Date;
PrtTime:=Time;
If PrtOpt = 'Y' Then Begin
   Writeln(Lst,Chr(15));
   Writeln(Lst,'* Monthly Failure Rates *');
   Writeln(Lst);
   PrtComments('N');
```

```
Writeln(Lst);
Writeln(Lst,'  Data File used for study: ',IFileName);
Writeln(Lst,'  Date of study:            ',PrtDate);
Writeln(Lst,'  Time of study:            ',PrtTime);
Writeln(Lst,^L);
For I:=1 to 12 Do Begin
   Write(Lst,'| Sqrd ',I:2,' = ',SqStrArr[I],'   || Sqrd ',I+12,' = ',SqStrArr[I + 12]);
   Write(Lst,'   || Sqrd ',I + 24,' = ');
   Writeln(Lst,SqStrArr[I + 24],'   || Sqrd  ',I+36,' = ',SqStrArr[I + 36],'   |');
End;
Writeln(Lst,^L);
If sq > 24 Then K:=2
Else K:=1;
For L:=1 to K Do Begin
  Write(Lst,'Number of aircraft as a function of time & sqd.              ');
  Writeln(Lst,'Sheet ':63,L:2,' of ',K:2);
  For I:=1 to 132 Do Write(Lst,'=');
  Writeln(Lst);
  Writeln(Lst,'  Date    |-------- Sqr. Number -------->');
  Write(Lst,'       ');

  If (Sq > 24) Then Begin
     If L = 1 Then Begin
        Start:=1; TheEnd:=24;
     End
     Else Begin
        Start:=25;  TheEnd:=sq;
     End
  End
  Else Begin
     Start:=1;  TheEnd:=sq;
  End;
  For I:=Start to TheEnd Do Write(Lst,I:5);
  If (((Sq > 24) and (L = 2)) or (Sq <=24)) Then
     Writeln(Lst,'    Tot.')
  Else Writeln(Lst);
  TQ:=0;
  For I:=1 to 48 Do Begin
      If MonthNo[I] MOD 12 <> 0 Then Begin
         PrtYr:=HoldYr + (MonthNo[I] DIV 12);
         ChkMonth:=MonthNo[I] MOD 12;
      End
      Else Begin
         PrtYr:=HoldYr + ((MonthNo[I] DIV 12) -1);
         ChkMonth:=12;
      End;
```

```
            Case ChkMonth of
               1: PrtMonth:='JAN. ';
               2: PrtMonth:='FEB. ';
               3: PrtMonth:='MAR. ';
               4: PrtMonth:='APR. ';
               5: PrtMonth:='MAY  ';
               6: PrtMonth:='JUNE ';
               7: PrtMonth:='JULY ';
               8: PrtMonth:='AUG. ';
               9: PrtMonth:='SEP. ';
              10: PrtMonth:='OCT. ';
              11: PrtMonth:='NOV. ';
              12: PrtMonth:='DEC. ';
            End;

            Write(Lst,PrtMonth,PrtYr:2);
            For J:=Start to TheEnd Do Begin
               If IntSq[I,J] <> 0 Then
                 Write(Lst,IntSq[I,J]:5:1)
               Else Write(Lst,'  ---');
               If IntSq[I,J] > 0 Then Begin
                  TQ:=TQ + 1;
                  Nsq[I]:=Nsq[I] + IntSq[I,J];
               End;
            End;
            If (((Sq > 24) and (L = 2)) or (Sq <=24)) Then
               Writeln(Lst,Nsq[I]:6:1)
            Else Writeln(Lst);
            TQ:=0;
            If Replace = -1 Then Nsq[I]:=Nsq[I] - Nco[I]
            Else If Replace = 1 Then Nsq[I]:=Nco[I] + Initial;
         End;
         Writeln(Lst,^L);
      End;
   End
   Else Begin
      Writeln(PrtFile,Chr(15));
      Writeln(PrtFile,'* Monthly Failure Rates *');
      Writeln(PrtFile);
      PrtComments('Y');
      Writeln(PrtFile);
      Writeln(PrtFile,'  Data File used for study: ',IFileName);
      Writeln(PrtFile,'  Date of study:            ',PrtDate);
      Writeln(PrtFile,'  Time of study:            ',PrtTime);
      Writeln(PrtFile,^L);
      For I:=1 to 12 Do Begin
```

```
   Write(PrtFile,'| Sqrd ',I:2,' = ',SqStrArr[I],'   || Sqrd ',I+12,' = ',SqStrArr[I + 12]);
   Write(PrtFile,'   || Sqrd ',I + 24,' = ');
   Writeln(PrtFile,SqStrArr[I + 24],'   || Sqrd  ',I+36,' = ',SqStrArr[I + 36],'   |');
End;
Writeln(PrtFile,^L);
If sq > 24 Then K:=2
Else K:=1;
For L:=1 to K Do Begin
   Write(PrtFile,'Number of aircraft as a function of time & sqd.              ');
   Writeln(PrtFile,'Sheet ':63,L:2,' of ',K:2);
   For I:=1 to 132 Do Write(PrtFile,'=');
   Writeln(PrtFile);
   Writeln(PrtFile,'  Date    |-------- Sqr. Number -------->');
   Write(PrtFile,'       ');

   If (Sq > 24) Then Begin
      If L = 1 Then Begin
         Start:=1; TheEnd:=24;
      End
      Else Begin
         Start:=25;  TheEnd:=sq;
      End
   End
   Else Begin
      Start:=1;  TheEnd:=sq;
   End;
   For I:=Start to TheEnd Do Write(PrtFile,I:5);
   If (((Sq > 24) and (L = 2)) or (Sq <=24)) Then
      Writeln(PrtFile,'    Tot.')
   Else Writeln(PrtFile);
   TQ:=0;
   For I:=1 to 48 Do Begin
       If MonthNo[I] MOD 12 <> 0 Then Begin
          PrtYr:=HoldYr + (MonthNo[I] DIV 12);
          ChkMonth:=MonthNo[I] MOD 12;
       End
       Else Begin
          PrtYr:=HoldYr + ((MonthNo[I] DIV 12) -1);
          ChkMonth:=12;
       End;
       Case ChkMonth of
          1: PrtMonth:='JAN. ';
          2: PrtMonth:='FEB. ';
          3: PrtMonth:='MAR. ';
          4: PrtMonth:='APR. ';
          5: PrtMonth:='MAY  ';
          6: PrtMonth:='JUNE ';
```

```
            7: PrtMonth:='JULY ';
            8: PrtMonth:='AUG. ';
            9: PrtMonth:='SEP. ';
           10: PrtMonth:='OCT. ';
           11: PrtMonth:='NOV. ';
           12: PrtMonth:='DEC. ';
          End;

          Write(PrtFile,PrtMonth,PrtYr:2);
          For J:=Start to TheEnd Do Begin
             If IntSq[I,J] <> 0 Then
                Write(PrtFile,IntSq[I,J]:5:1)
             Else Write(PrtFile,'  ---');
             If IntSq[I,J] > 0 Then Begin
                TQ:=TQ + 1;
                Nsq[I]:=Nsq[I] + IntSq[I,J];
             End;
          End;
          If (((Sq > 24) and (L = 2)) or (Sq <=24)) Then
             Writeln(PrtFile,Nsq[I]:6:1)
          Else Writeln(PrtFile);
          TQ:=0;
          If Replace = -1 Then Nsq[I]:=Nsq[I] - Nco[I]
          Else If Replace = 1 Then Nsq[I]:=Nco[I] + Initial;
       End;
       Writeln(PrtFile,^L);
    End;
End;
Close(Sqrdqt);
T1:=99999.0; T2:=0; DT:=0;
N1:=0; N2:=0; AR:=0;
For I:=1 to 50 Do
     For J:=1 to 50 Do
          Fsq[I,J]:=0.0;
While Not EOF(InFile) Do Begin
      Readln(InFile,Line);
      BN:=Copy(Line,2,6);
      TStr:=Copy(Line,56,7);
      I:=1;
      Repeat
        If TStr[I] = ' ' Then Delete(TStr,I,1)
        Else I:=I + 1;
      Until (I >= Length(TStr));
      Val(TStr,T,Code);
      ii:=Trunc(T/30.4) + 1;
      ii:=(ii - BegMonth) + 1;
      If T > T2 Then T2:=T;
```

```
      If ii > 50 Then ii:=50;
      ChkSqd:=Copy(Line,53,3);
      I:=1;  Found:=False;
      While ((I <=Sq) and (Found = False)) Do Begin
          If ChkSqd = SqStrArr[I] Then Begin
              Fsq[ii,I]:=Fsq[ii,I] + 1;
              N[ii]:=N[ii] + 1;
              Found:=True;
          End;
          I:=I + 1;
      End;
      N1:=N1 + 1;
      If T < T1 Then T1:=T;
      FStr:=Copy(Line,129,6);
      I:=1;
      Repeat
        If FStr[I] = ' ' Then Delete(FStr,I,1)
        Else I:=I + 1;
      Until (I >= Length(FStr));
      Val(FStr,F,Code);
      If F > 0 Then DT:=DT + (T - F);
      NpStr:=Copy(Line,135,3);
      K:=Pos('.',NpStr);
      If ((K = 1) or ((K > 1) and (NpStr[K - 1] = ' '))) Then
        Insert('0',NpStr,K);
      I:=1;
      Repeat
        If NpStr[I] = ' ' Then Delete(NpStr,I,1)
        Else I:=I + 1;
      Until (I >= Length(NpStr));
      Val(NpStr,np,Code);
      If np = 0 Then AR:=AR + 1;
End;
N2:=AR;
For J:=1 to 48 Do
    For I:=1 to 48 Do
        If IntSQ[I,J] <> 0 Then Av[J]:=Av[J] + 1;

For J:=1 to 48 Do Begin
    Xsq[J]:=0;
    For I:=1 to Sq Do
      Xsq[J]:=Xsq[J] + Fsq[J,I];
End;
If PrtOpt = 'Y' Then Begin
   PrtTitle:='Number of failures as a function of time & sqd.            ';
   PrintValPrt;
End
```

```
Else Begin
   PrtTitle:='Number of failures as a function of time & sqd.            ';
   PrintValFile;
End;
For I:=1 to 48 Do  Begin
    For J:=1 to 48 Do Begin
        Isb:=IntSq[I,J]; If Isb = 0 Then Isb:=1;
        Fsq[I,J]:=Fsq[I,J]/Isb;
    End;
End;
For I:=1 to 48 Do Begin
    Bot:=Nsq[I]; If Bot = 0 Then Bot:=1;
    Xsq[I]:=N[I]/Bot;
End;
If PrtOpt = 'Y' Then Begin
   PrtTitle:='Number of failures per aircraft as a function of time & sqd.';
   PrintValPrt;
End
Else Begin
   PrtTitle:='Number of failures per aircraft as a function of time & sqd.';
   PrintValFile;
End;

Mtbf:= (T2 - T1) * N2/N1;
DDT:=(T2 - T1)/48;
TT:=Mtbf - DT/N1;
If N2 > 0 Then TT:=TT*N1/N2;
Ta:=TT;
Max:=1.5 * Ta;
ClrScr;
For K:=1 to 2 Do Begin
    If K = 1 Then
       I:=Trunc(T2/30.4) + 1
    Else I:=Trunc(T1/30.4) + 1;
    If I MOD 12 <> 0 Then Begin
        PrtYr:=HoldYr + (I DIV 12);
        ChkMonth:=I MOD 12;
    End
    Else Begin
        PrtYr:=HoldYr + ((I DIV 12) -1);
        ChkMonth:=12;
    End;
    Case ChkMonth of
        1: PrtMonth:='JAN. ';
        2: PrtMonth:='FEB. ';
        3: PrtMonth:='MAR. ';
        4: PrtMonth:='APR. ';
```

```
        5: PrtMonth:='MAY  ';
        6: PrtMonth:='JUNE ';
        7: PrtMonth:='JULY ';
        8: PrtMonth:='AUG. ';
        9: PrtMonth:='SEP. ';
       10: PrtMonth:='OCT. ';
       11: PrtMonth:='NOV. ';
       12: PrtMonth:='DEC. ';
    End;
    If K = 1 Then Begin
       PrtY1:=PrtYr;
       PrtM1:=PrtMonth;
    End
    Else Begin
       PrtY2:=PrtYr;
       PrtM2:=PrtMonth;
    End;
End;
GotoXY(1,1);
Writeln('Max time in data        =',T2:6:0,'   ',PrtM1,PrtY1:2);
Writeln('Min time in data        =',T1:6:0,'   ',PrtM2,PrtY2:2);
Writeln('Total number of data    =',N1:6:0);
Writeln('Number to be cal.       =',N2:6:0);
Writeln('Mean time between fail.=',Mtbf:10:3);
Writeln('Mean value for unseen   =',Ta:10:3);
Writeln('Max time for cal. time =',Max:10:3);

If PrtOpt = 'Y' Then Begin
   Writeln(Lst,'Max time in data        =',T2:6:0,'   ',PrtM1,PrtY1:2);
   Writeln(Lst,'Min time in data        =',T1:6:0,'   ',PrtM2,PrtY2:2);
   Writeln(Lst,'Total number of data    =',N1:6:0);
   Writeln(Lst,'Number to be cal.       =',N2:6:0);
   Writeln(Lst,'Mean time between fail.=',Mtbf:10:3);
   Writeln(Lst,'Mean value for unseen   =',Ta:10:3);
   Writeln(Lst,'Max time for cal. time =',Max:10:3);
   Writeln(Lst,^L);
End
Else Begin
   Writeln(PrtFile,'Max time in data        =',T2:6:0,'   ',PrtM1,PrtY1:2);
   Writeln(PrtFile,'Min time in data        =',T1:6:0,'   ',Prtm2,PrtY2:2);
   Writeln(PrtFile,'Total number of data    =',N1:6:0);
   Writeln(PrtFile,'Number to be cal.       =',N2:6:0);
   Writeln(PrtFile,'Mean time between fail.=',Mtbf:10:3);
   Writeln(PrtFile,'Mean value for unseen   =',Ta:10:3);
   Writeln(PrtFile,'Max time for cal. time =',Max:10:3);
   Writeln(PrtFile,^L);
End;
```

```
Close(InFile);
Assign(InFile,'Mtbf.fac');
Rewrite(InFile);
Nt:=0;  Ns:=0;  Nv:=0;  Nv2:=0; Nvd:=0;  Ncd:=0;  Se:=0;  Sev:=0;
I1:= 1; I48:=12;  WibelFit;  AA[1]:=A;  BB[1]:=B;
I1:=13; I48:=24;  WibelFit;  AA[2]:=A;  BB[2]:=B;
I1:=25; I48:=36;  WibelFit;  AA[3]:=A;  BB[3]:=B;
I1:=37; I48:=48;  WibelFit;  AA[4]:=A;  BB[4]:=B;
I1:= 1; I48:=48;  WibelFit;  AA[5]:=A;  BB[5]:=B;
If PrtOpt = 'Y' Then Begin
   Writeln(Lst,^L);
   Writeln(Lst,'* Monthly Failure Rates *');
   Writeln(Lst);
   Writeln(Lst);
   Writeln(Lst,'  Data File used for study: ',IFileName);
   Writeln(Lst,'  Date of study:            ',PrtDate);
   Writeln(Lst);
   Writeln(Lst,' Date     Num. Adv. Adv. Ratio of Cal.  Delta  Var. Delta');
   Writeln(Lst,'          Fail Inv. Tbf  Tbf/MTBF Tbf   cal-T  Tbf  var-T');
   Writeln(Lst,'-------   ---- ---- ---- -------- ----  ----- ----  -----');
End
Else Begin
   Writeln(PrtFile,^L);
   Writeln(PrtFile,'* Monthly Failure Rates *');
   Writeln(PrtFile);
   Writeln(PrtFile);
   Writeln(PrtFile,'  Data File used for study: ',IFileName);
   Writeln(PrtFile,'  Date of study:            ',PrtDate);
   Writeln(PrtFile);
   Writeln(PrtFile,' Date     Num. Adv. Adv. Ratio of Cal.  Delta  Var. Delta');
   Writeln(PrtFile,'          Fail Inv. Tbf  Tbf/MTBF Tbf   cal-T  Tbf  var-T');
   Writeln(PrtFile,'-------   ---- ---- ---- -------- ----  ----- ----  -----');
End;
For I:=1 to 48 Do Begin
    V:=Fdt[I];  V2:=V/Mtbf;  Nt:=Nt + N[I];Ns:=Ns + Nsq[I];  Nv:=Nv + V;
    Nv2:=Nv2 + V2;
    Tv:=I * 30.4;
    Cdt:=A * (Exp(b * ln(tv)));  dc:=cdt - v;  se:=se + dc*dc;
    ix:=1 + Trunc(I/13.0);  Vdt:=aa[ix] * (Exp(bb[ix] * ln(tv)));
    dv:=vdt - v; sev:=sev + dv* dv;
    Nvd:=Nvd + vdt;
    Ncd:=Ncd + cdt;
    If MonthNo[I] MOD 12 <> 0 Then Begin
       PrtYr:=HoldYr + (MonthNo[I] DIV 12);
       ChkMonth:=MonthNo[I] MOD 12;
    End
```

```
        Else Begin
           PrtYr:=HoldYr + ((MonthNo[I] DIV 12) -1);
           ChkMonth:=12;
        End;
        Case ChkMonth of
           1: PrtMonth:='JAN. ';
           2: PrtMonth:='FEB. ';
           3: PrtMonth:='MAR. ';
           4: PrtMonth:='APR. ';
           5: PrtMonth:='MAY  ';
           6: PrtMonth:='JUNE ';
           7: PrtMonth:='JULY ';
           8: PrtMonth:='AUG. ';
           9: PrtMonth:='SEP. ';
          10: PrtMonth:='OCT. ';
          11: PrtMonth:='NOV. ';
          12: PrtMonth:='DEC. ';
        End;

        If PrtOpt = 'Y' Then
           Writeln(Lst,PrtMonth,PrtYr:2,n[I]:7,Nsq[I]:5:0,V:5:0,V2:7:2,cdt:7:0,dc:7:1,vdt:5:0,dv:7:1)
        Else
           Writeln(PrtFile,PrtMonth,PrtYr:2,n[I]:7,Nsq[I]:5:0,V:5:0,V2:7:2,cdt:7:0,dc:7:1,vdt:5:0,dv:7:1);
        Writeln(InFile,V2);

     End;
     If PrtOpt = 'Y' Then Begin
        Writeln(Lst,'          ____________________________________________________________________________________________

Writeln(Lst,Nt:14:0,Ns/48:5:0,Nv/48:5:0,Nv2/48:7:2,ncd/48:7:0,Sqrt(Se/47):7:1,Nvd/48:5:0,Sqrt(sev/47):7:1);
        Writeln(Lst,'          ==============================================');
     End
     Else Begin
        Writeln(PrtFile,'          ____________________________________________________________________________________________

Writeln(PrtFile,Nt:14:0,Ns/48:5:0,Nv/48:5:0,Nv2/48:7:2,ncd/48:7:0,Sqrt(Se/47):7:1,Nvd/48:5:0,Sqrt(sev/47):
        Writeln(PrtFile,'          ==============================================');
        Flush(PrtFile);
        Close(PrtFile);
     End;
     Close(InFile);
     TempCom:='MONTHLY FAILURE REPORT CREATED USING INPUT FILE ' + IFileName;
     UpdateCom;
  End
  Else Begin
     ClrScr;
```

```
      GotoXY(15,12);
      Write(^G,'Sqrdqt.sqr does not exist');
    End;

End.    (* Program MonthFail *)^Z
```

[BLANK PAGE]

Appendix J.26

PrtComments

```
Procedure PrtComments(PrtOpt:Char);

TYPE
   BigStr = String[255];

VAR
  CFile: Text;
  CName: Path;
  CH: Char;
  Found: Boolean;
  Row,I: Integer;
  NoChar,StartN,EndN: Integer;
  Image: ImageType;
  Name: String20;
  TempCom,GenComments,DirCom,UserCom: BigStr;

Begin  (* Procedure Comments *)

   CName:='DataList.cmt';
   If DrvDir <> ProgDrvDir Then AddDir(CName);
   Assign(CFile,CName);
   Reset(CFile);
   Readln(CFile,GenComments);
   Close(CFile);
   TempCom:=GenComments;
   (* Break up Comments *)
   NoChar:=Length(TempCom);
   EndN:=0;
   While (EndN + 1 <=NoChar) Do Begin
      StartN:=EndN + 1; Found:=False;
      While ((StartN <=NoChar) and (Found = False)) Do Begin
          If TempCom[StartN] = ' ' Then StartN:=StartN + 1
          Else Found:=True;
      End;
      EndN:=StartN + 51;
      If EndN > NoChar Then EndN:=NoChar;
      (* Break up into words *)
      Found:=False;
      If ((EndN <> NoChar) and (TempCom[EndN + 1] <> ' ')) Then
         While ((EndN > StartN) and (Found = False)) Do Begin
             If TempCom[EndN] <> ' ' Then
                EndN:=EndN - 1
             Else Found:=True;
```

```
        End;
      If PrtOpt = 'Y' Then Begin
        For I:=StartN to EndN Do Write(PrtFile,TempCom[I]);
        Writeln(PrtFile);
      End
      Else Begin
        For I:=StartN to EndN Do Write(Lst,TempCom[I]);
        Writeln(Lst);
      End;
   End;
End;   (* Procedure Comments *)
^Z
```

Appendix J.27

ReadCom

```
Procedure ReadCom;

 VAR
   SendArr: Path;

Begin   (* Procedure ReadCom *)

  Quit:=False; Field:=1; MaxField:=5;  UppCase:=True;
  HlpMsg1:='Update/Enter Comments concerning the file:';
  HlpMsg2:=OFileName;
  While ((Field <= MaxField) and (Quit = False)) Do Begin
     SendArr:='';
     Case Field of
        1: Begin
             For I:=1 to 51 Do SendArr:=SendArr + ' ';
             For I:=1 to Length(TmpCom1) Do SendArr[I]:=TmpCom1[I];
             ReadField(SendArr,15,11,51);
             TmpCom1:=Copy(SendArr,1,51);
           End;
        2: Begin
             For I:=1 to 51 Do SendArr:=SendArr + ' ';
             For I:=1 to Length(TmpCom2) Do SendArr[I]:=TmpCom2[I];
             ReadField(SendArr,15,12,51);
             TmpCom2:=Copy(SendArr,1,51);
           End;
        3: Begin
             For I:=1 to 51 Do SendArr:=SendArr + ' ';
             For I:=1 to Length(TmpCom3) Do SendArr[I]:=TmpCom3[I];
             ReadField(SendArr,15,13,51);
             TmpCom3:=Copy(SendArr,1,51);
           End;
        4: Begin
             For I:=1 to 51 Do SendArr:=SendArr + ' ';
             For I:=1 to Length(TmpCom4) Do SendArr[I]:=TmpCom4[I];
             ReadField(SendArr,15,14,51);
             TmpCom4:=Copy(SendArr,1,51);
           End;
        5: Begin
             For I:=1 to 51 Do SendArr:=SendArr + ' ';
             For I:=1 to Length(TmpCom5) Do SendArr[I]:=TmpCom5[I];
             ReadField(SendArr,15,15,51);
```

```
            TmpCom5:=Copy(SendArr,1,51);
          End;
      End;  (* CaseStatement *)

    End;  (* While Loop *)
  End;   (* Procedure ReadCom *)

(*=================================================================*)
  Procedure BuildCom;

  (*=================================================================*)
     Procedure FindEnd(VAR HoldStr:Path);

      VAR
        I: integer;

     Begin (* Procedure FindEnd *)

        I:=Length(HoldStr);
        While HoldStr[I] = ' ' Do I:=I - 1;
        Delete(HoldStr,I + 1,51 - I);

     End;  (* Procedure FindEnd *)
  (*=================================================================*)

  Begin  (* Procedure BuildCom *)

     If Length(TmpCom1) > 1 Then FindEnd(TmpCom1);
     If Length(TmpCom2) > 1 Then FindEnd(TmpCom2);
     If Length(TmpCom3) > 1 Then FindEnd(TmpCom3);
     If Length(TmpCom4) > 1 Then FindEnd(TmpCom4);
     If Length(TmpCom5) > 1 Then FindEnd(TmpCom5);
     TempCom:='';
     TempCom:=TmpCom1 + ' ' + TmpCom2 + ' ' + TmpCom3 + ' ' + TmpCom4 + ' ' + TmpCom5;

  End;  (* Procedure BuildCom *)
^Z
```

Appendix J.28

ReadField

Procedure ReadField(Var Arrsent:Path;Col,Row,Limit:integer);

```
Screen control to read in data for the specified field
ArrSent contains the value of the fields, row and col
designate where to go on the screen, limit holds the
length of the field, Skip1 and Skip2 indicate where the
cursor should skip over, [example: date field ___/___/___,
the cursor should skip over the slashes]
```

```
Var
  Hold: Array [1..64] of char;
  Ch:char;
  EndCol,HoldCol:integer;
  I,J,K:integer;
  Continue:Boolean;

Begin (* Procedure ReadField *)
  For I:=1 to 64 Do Hold[I]:=' ';
  For I:=1 to Length(ArrSent)  Do Hold[I]:=ArrSent[I];
  I:=1;
  HoldCol:=Col;
  EndCol:=Col + Limit - 1;
  Continue:=True;
  ch:=' ';
  While ((Ord(ch) <> 13) and (Continue = True))
  Do Begin
     GotoXY(Col,Row); Read(Kbd,ch);
     Case Ord(Ch) of
            27: Begin
                  If KeyPressed Then Begin
                    Read(Kbd,Ch);
                    Case Ord(Ch) of
         (* F1=>Help *)  59: Begin
                              Help(HlpMsg1,HlpMsg2);
                            End;
   (* 15 Tab Left *)  15,72: Begin
   (* 72 Up Arrow *)          If Field > 1 Then
                                 Field:=Field - 1;
                              Continue:=False;
                            End;
```

```
        (* Down Arrow *)  80: Begin
                                If Field <= MaxField  Then Field:=Field + 1;
                                Continue:=False;
                               End;
        (* Left Arrow *)  75: If Col > HoldCol Then Begin
                                 Col:=Col - 1;
                                 I:= I - 1;
                               End;
       (* Right Arrow *)  77: Begin
                                If I > Limit Then Begin
                                   GotoXY(Col,Row); Write(^G);
                                End
                                Else If I <= Limit Then Begin
                                   Col:=Col + 1;
                                   I:= I + 1;
                                End;
                              End;
     (* Delete Key *)     83: If (I <=Limit) Then Begin
                                 If (I < Limit) Then
                                     For J:=I to Limit - 1 Do Hold[J]:=Hold[J + 1];
                                 Hold[Limit]:=' ';
                                 GotoXY(Col,Row);
                                 For J:=I to Limit Do Begin
                                     If (Hold[J] <> ' ') Then Write(Hold[J])
                                     Else Write('_');
                                 End;
                              End;
                         End; (* Case Statement *)
                     End  (* KeyPressed *)
                     Else Begin
                         Quit:=True;
                         Continue:=False;
                     End;
                    End;  (* Ord = 27 *)
(* Tab Right *)  9: Begin
                       If Field <= MaxField  Then Field:=Field + 1;
                       Continue:=False;
                     End;
(* Char *) 32..125: Begin
                       If I > Limit Then Begin
                          GotoXY(Col,Row);
                          Write(^G);
                       End
                       Else If I <= Limit Then Begin
                           If UppCase = True Then Ch:=UpCase(Ch);
                           Hold[I]:=ch;
                           GotoXY(Col,Row); Write(ch);
```

```
                        I:=I+1;
                        Col:=Col+1;
                    End;
                End;
  (* Return *) 13: If Field <= MaxField Then Field:=Field + 1;
(*Back Arrow *)  8: If (I > 1) Then Begin
                        I:=I - 1;
                        For J:=I to Limit - 1 Do Hold[J]:=Hold[J + 1];
                        Hold[Limit]:=' ';
                        Col:=Col - 1;
                        GotoXY(Col,Row);
                        For J:=I to Limit Do Begin
                            If (Hold[J] <> ' ') Then Write(Hold[J])
                            Else Write('_');
                        End;
                    End;
            End; (* of Case *)
        End; (* of While *)
        ArrSent:='';
        While Hold[Limit] = ' ' Do Limit:=Limit - 1;
        For I:=1 to Limit Do ArrSent:=ArrSent + Hold[I];

    End;  (* Procedure ReadField *)

^Z
```

[BLANK PAGE]

Appendix J.29

RegraAnl

```
Program RegrAnl(Input,Output);
{$C-}
```

STEP 4 - Regression Analysis. This step performs linear regression analysis on the file generated by Step 3. The correlation equation is assumed to be linear and each parameter can be omitted to determine its effect. This program can also generate plots of data.

Deborah A. Holder

```
CONST
   F1Min = 0;
   F1Max = 10000;

TYPE
   HoldName = array [1..25] of char;
   DateTimeType = String[8];
   RegRec = record
              ax,bx,cx,dx,bp,si,di,ds,es,flags: integer;
            end;
   String255 = String[255];
   String50 = String[50];
   String10 = string[10];
   String20 = string[20];
   String3  = string[3];
   Char12arr = array [1..12] of char;
   String6 = string[6];
   DPointer = ^DirRec;
   DirRec = record
              Size: real;
              Hr,Minu: integer;
              Year,Month,Day: integer;
              Next,Prev: DPointer;
              Name: String20;
            end;
   APointer = ^AnalysisRec;
   AnalysisRec = record
               F:real;
               VX: array [1..6] of real;
```

```
            FO1: real;
            FO2: real;
            Next: APointer;
          end;

  TenRealNo = array [1..10] of real;
  ImageType = array[1..4096] of char;
  Path = string[64];

VAR
  HeapTop: ^Integer;
  NoFiles: integer;
  Top,Bottom: DPointer;
  FirstRec,LastRec,NewRec: DPointer;
  FirstARec,LastARec: APointer;
  Track,NewARec: APointer;
  FPName:String[25];
  OFileName,PFileName,IFileName: Path;
  GFN,PFile,PrtFile,InFile: Text;
  KeptTrack,Omit,Counter: integer;
  PrtDate,PrtTime: DateTimeType;
  F1S,F1: real;
  RVX: array [1..6] of real;
  UppCase,Default,Quit,ReqFile,ValFile: Boolean;
  Code,I,J,K,L,IJP,ILP,Num,NumRec,NumVar,NoVar: integer;
  NovT,NovP: integer;
  NumStr: string6;
  RH,XX,VS: TenRealNo;
  JO: array [1..11] of integer;
  C: array [1..10] of TenRealNo;
  Sum,FA,FB,T1,T2,BV1,AV1: real;
  DF,FV1,DF1,F1A,SDV1,SD2: real;
  SSA,SS1,SS12,SS2: real;
  ZC1,ZC2: real;
  Ans,PrtOpt: char;
  Field,MaxField: Integer;
  Image: ImageType;
  CrtMode: Byte absolute $0040:$0049;
  MonoBuffer: ImageType absolute $B000:$0000;
  ColorBuffer: ImageType absolute $B800:$0000;
  ChkFile,ChkDir,ProgDrvDir,DrvDir: Path;
  TempCom: String255;
  RdStr,TmpCom1,TmpCom2,TmpCom3,TmpCom4,TmpCom5: Path;
  HlpMsg1,HlpMsg2: String255;
```

```
{$IBoxWin.pas}
{$IHelp.pas}
{$IAddDir.pas}
{$IComments.pas}
{$IExists.pas}
{$IDiskSpace.pas}
{$IGetDir.pas}
{$IDirLst.pas}
{$IFileName.pas}
{$IAddFileExt.pas}
{$IDate.pas}
{$ITime.pas}
{$IPrtComments.pas}
{$IReadField.pas}
{$IReadCom.pas}
{$IDisCom.pas}
{$IUpdateCom.pas}

(*==========================================================================*)
  Procedure InvertMatrix;

   VAR
     J,J1,K,I: integer;
     CC,SO,CO: real;

  Begin   (* Procedure InvertMatrix *)

     For I:=1 to NumVar + 1 Do
        JO[I]:=I;

     For I:=1 to NumVar + 1 Do Begin
         CO:=0;
         J1:=I;
         For J:=I to NumVar + 1 Do Begin
             If (Abs(CO) - Abs(C[I,J])) < 0 Then Begin
                J1:=J;
                CO:=C[I,J];
             End;
         End;
         If I <> J1 Then Begin
            K:=JO[J1];
            JO[J1]:=JO[I]; JO[I]:=K;
            For K:=1 to NumVar + 1 Do Begin
                SO:=C[K,I];  C[K,I]:=C[K,J1];
                C[K,J1]:=SO;
            End;
         End;
```

```
        C[I,I]:=1;
        For J1:=1 to NumVar + 1 Do Begin
           CC:=C[I,J1];
           If CO <> 0 Then
              CC:=CC/CO;
           C[I,J1]:=CC;
        End;
        For J1:=1 to NumVar + 1 Do Begin
           If I <> J1 Then Begin
              CO:=C[J1,I];
              If CO <> 0 Then Begin
                 C[J1,I]:=0;
                 For K:=1 to NumVar + 1 Do
                     C[J1,K]:=C[J1,K] - CO * C[I,K];
              End;
           End;
        End;
     End;

     For I:=1 to NumVar + 1 Do Begin
        If JO[I] <> I Then Begin
           J1:=I;
           Repeat
             J1:=J1 + 1;
           Until ((JO[J1] = I) or (NumVar + 1 <= J1));
           JO[J1]:=JO[I];
           For K:=1 to NumVar + 1 Do Begin
              CO:=C[I,K];  C[I,K]:=C[J1,K];
              C[J1,K]:=CO;
           End;
           JO[I]:=I;
        End;
     End;

  End;    (* Procedure InvertMatrix *)
(*=====================================================================*)

Begin  (* Program RegrAnl *)

    GraphBackGround(Blue);
    TextBackGround(Blue);
    TextColor(Yellow);
    ClrScr;
    Default:=False;
    BoxWin(1,1,80,24,'  STEP 4  ');
    GotoXY(25,10);
    Write('>>> Regression Analysis <<<');
```

```
GotoXY(1,1);
GetDir(0,ProgDrvDir);
Assign(GFN,'GFN_Nail');
Reset(GFN);
ChkFile:=''; ChkDir:=''; DrvDir:='';
Readln(GFN,DrvDir);
Readln(Gfn,ChkFile);
Close(GFN);
ChkDir:=DrvDir;
If Length(ChkDir) = 2 Then ChkDir:=ChkDir + '\';
Delay(1000);
If ((ChkFile <> '**') and (Pos('DTD',ChkFile) > 0)) Then Begin
   Default:=True;
   IFileName:=''; IFileName:=ChkFile;
   If DrvDir <> ProgDrvDir Then AddDir(IFileName);
End
Else Begin
   Mark(HeapTop);
   GetDirectory('????????.dtd');

   ValFile:=False;
   ReqFile:=True;
   While ValFile = False Do Begin
       IFileName:='                    ';
       FileName(IFileName,' Input   ');
       AddFileExt(IFileName,'DTD');
       If DrvDir <> ProgDrvDir Then AddDir(IFileName);
       ValFile:=Exists(IFileName);
       If ValFile = False Then Begin
          GotoXY(1,12); ClrEOL;
          GotoXY(15,12); Write(^G,'File Does Not Exist');
          Delay(3000);
       End;
   End;

   Release(HeapTop);
End;
Assign(InFile,IFileName);
ReSet(InFile);

If Default = False Then Begin
   ClrScr;
   GotoXY(15,10);
   NumStr:=''; NumRec:=0;
   Write('NUMBER OF X/Y DATA (Top) =: ');
   MaxField:=1;  Field:=1;  Quit:=False;
   RdStr:='';
```

```
   HlpMsg1:='Enter the number of records you wish to analysis.';
   HlpMsg2:='';
   ReadField(RdStr,WhereX,WhereY,3);
   Val(RdStr,NumRec,Code);
   NumStr:=''; NoVar:=0;
   GotoXY(15,12);
   Write('NUMBER OF VARIABLES      =: ');
   MaxField:=1;  Field:=1;  Quit:=False;
   RdStr:='';
   HlpMsg1:='Enter the number of variables per record.';
   HlpMsg2:='';
   ReadField(RdStr,WhereX,WhereY,3);
   Val(RdStr,NoVar,Code);
   NumStr:=''; Omit:=0;
   GotoXY(15,14);
   Write('OMIT VARIABLE NUMBER      =: ');
   MaxField:=1;  Field:=1;  Quit:=False;
   RdStr:='';
   HlpMsg1:='Enter the variable number you wish to omit, if any.';
   HlpMsg2:='';
   ReadField(RdStr,WhereX,WhereY,3);
   Val(RdStr,Omit,Code);
   GotoXY(15,16);
   Write('Output: System Printer [Y/N] : ');
   GotoXY(WhereX,WhereY);
   Read(kbd,PrtOpt);  PrtOpt:=UpCase(PrtOpt);
   GotoXY(WhereX,WhereY); Write(PrtOpt);
End
Else Begin
   PrtOpt:='N';
   NoVar:=20;
   NumRec:=4000;
   Omit:=0;
End;
If PrtOpt <> 'Y' Then Begin
   If ((ChkFile <> '**') and (Pos('DTD',ChkFile) > 0)) Then Begin
      OFileName:=''; OFileName:=ChkFile;
      AddFileExt(OFileName,'R04');
      If DrvDir <> ProgDrvDir Then AddDir(OFileName);
   End
   Else Begin
      Mark(HeapTop);
      GetDirectory('????????.R04');
      ValFile:=True;
      While ValFile = True Do Begin
            ReqFile:=True;
            OFileName:='                    ';
```

```
            FileName(OFileName,'  Report  ');
            AddFileExt(OFileName,'R04');
            If DrvDir <> ProgDrvDir Then AddDir(OFileName);
            ValFile:=Exists(OFileName);
            If ValFile = True Then Begin
               GotoXY(1,12); ClrEOL;
               GotoXY(15,12); Write(^G,'File Already Exist !!');
               GotoXY(15,13); Write('Do you wish to overwrite?: __');
               GotoXY(WhereX -1,WhereY);
               Read(Kbd,Ans);
               Ans:=UpCase(Ans);
               GotoXY(WhereX -1,WhereY); Write(Ans);
               If Ans = 'Y' Then ValFile:=False;
            End;
         End;
         Release(HeapTop);
      End;
      Assign(PrtFile,OFileName);
      Rewrite(PrtFile);
   End;

   Mark(HeapTop);
   For J:=1 to 10 Do Begin
      For I:=1 to 10 Do C[I,J]:=0;
      JO[J]:=0; RH[J]:=0; XX[J]:=0; VS[J]:=0;
   End;
   JO[11]:=0;
   Counter:=1;
   FirstARec:=Nil;
   F1S:=0;
   Readln(InFile,NumVar); If NoVar < NumVar Then NumVar:=NoVar;
   While ((NOT EOF(InFile))  and (Counter <= NumRec))Do Begin
      KeptTrack:=Counter;
      Readln(InFile,F1);
      For J:=1 to NumVar Do
          Readln(InFile,RVX[J]);
      If ((F1 > F1Min) and (F1 < F1Max)) Then
         (* Do Nothing *)
      Else Begin
         F1:=F1S;
         For J:=1 to NumVar Do
            RVX[J]:=VS[J];
      End;
      F1S:=F1;
      For J:=1 to NumVar Do Begin
          If Omit = J Then RVX[J]:=0;
          VS[J]:=RVX[J];
      End;
```

```
    RH[1]:=RH[1]+ F1;
    For J:=1 to NumVar Do
        RH[J + 1]:=RH[J + 1] + F1 * VS[J];
    C[1,1]:=C[1,1] + 1;
    For J:=1 to NumVar Do Begin
       C[J + 1,1]:=C[J + 1,1] + VS[J];
       C[1,J + 1]:=C[J + 1,1];
       IJP:=J + 1;
       For L:=1 to J Do Begin
           ILP:=L + 1;
           C[IJP,ILP]:=C[IJP,ILP] + VS[J] * VS[L];
           C[ILP,IJP]:=C[IJP,ILP];
       End;
    End;
    New(NewARec);
    NewARec^.F:=F1;
    For J:=1 to NumVar Do
        NewARec^.VX[J]:=RVX[J];
    If FirstARec = Nil Then
       FirstARec:=NewARec
    Else LastARec^.Next:=NewARec;
    LastARec:=NewARec;
    LastARec^.Next:=Nil;
    Counter:=Counter + 1;
End;
Close(InFile);
T1:=RH[1] * C[2,1] - C[1,1] * RH[2];
T2:=C[2,1] * C[2,1] - C[1,1] * C[2,2];
BV1:=0;
If T2 <> 0 Then BV1:=T1/T2;
AV1:=(RH[1] - BV1 * C[2,1]) / C[1,1];

ClrScr;
Writeln;
K:=NumVar + 1;
If K > 6 Then K:=6;
For J:=1 to NumVar + 1 Do Begin
    For I:=1 to K Do
        Write(C[J,I]:9,'  ');
    Writeln('|',RH[J]:7:2);
End;
InvertMatrix;
For I:=1 to NumVar + 1 Do Begin
    Sum:=0;
    For J:=1 to NumVar + 1 Do
       Sum:=Sum + C[I,J] * RH[J];
```

```
    VS[I]:=0;
    XX[I]:=Sum;
End;
Track:=FirstARec;
SDV1:=0;  F1A:=0;
PrtDate:=Date;
PrtTime:=Time;
FA:=0;   FB:=0;  SD2:=0;
NovT:=NumVar; If NovT > 4 Then NovT:=4;
NovP:=NumVar; If NovP > 9 Then NovT:=9;
Writeln(' No.     F     f12     df12     f1     df1     V1     V2   ...');
If PrtOpt = 'Y' Then Begin
   Writeln(Lst,Chr(15));
   Writeln(Lst,'* Regression Analysis *');
   Writeln(Lst);
   PrtComments('N');
   Writeln(Lst);
   Writeln(Lst,'  Data File used for study: ',IFileName);
   Writeln(Lst,'  Date of study:            ',PrtDate);
   Writeln(Lst,'  Time of study:            ',PrtTime);
   Writeln(Lst,'  Number of X/Y Data (Top): ',Counter - 1:4);
   Writeln(Lst,'  Number of Variables:      ',NumVar:4);
   Writeln(Lst,^L);
   Writeln(Lst,'* Regression Analysis *');
   Writeln(Lst);
   Writeln(Lst,'  Data File used for study: ',IFileName);
   Writeln(Lst,'  Date of study:            ',PrtDate);
   Writeln(Lst);
   Writeln(Lst,' No.     F     f12     df12     f1     df1     V1     V2   ...');
End
Else Begin
   Writeln(PrtFile,Chr(15));
   Writeln(PrtFile,'* Regression Analysis *');
   Writeln(PrtFile);
   PrtComments('Y');
   Writeln(PrtFile);
   Writeln(PrtFile,'  Data File used for study: ',IFileName);
   Writeln(PrtFile,'  Date of study:            ',PrtDate);
   Writeln(PrtFile,'  Time of study:            ',PrtTime);
   Writeln(PrtFile,'  Number of X/Y Data (Top): ',Counter - 1:4);
   Writeln(PrtFile,'  Number of Variables:      ',NumVar:4);
   Writeln(PrtFile,^L);
   Writeln(PrtFile,'* Regression Analysis *');
   Writeln(PrtFile);
   Writeln(PrtFile,'  Data File used for study: ',IFileName);
   Writeln(PrtFile,'  Date of study:            ',PrtDate);
   Writeln(PrtFile);
   Writeln(PrtFile,' No.     F     f12     df12     f1     df1     V1     V2   ...');
```

```
End;
Counter:=1;
Repeat
  F1:=XX[1];
  For J:=1 to NumVar Do
      F1:=F1 + XX[J + 1] * Track^.VX[J];
  DF:=Track^.F - F1;
  FV1:=AV1 + BV1 * Track^.VX[1];
  DF1:=Track^.F - FV1;
  SDV1:=SDV1 + DF1 * DF1;
  F1A:=F1A + FV1;
  SD2:=SD2 + DF * DF;
  FA:=FA + Track^.F;
  FB:=FB + F1;
  For J:=1 to NumVar + 1 Do
      VS[J]:=VS[J] + Track^.VX[J];
  Write(Counter:3,Track^.F:9:1,F1:7:0,DF:8:0,FV1:8:0,DF1:9:1);
  For I:=1 to NovT Do
      Write(Track^.VX[I]:8:1);
  Writeln;
  If PrtOpt = 'Y' Then Begin
     Write(Lst,Counter:3,Track^.F:9:1,F1:7:0,DF:8:0,FV1:8:0,DF1:9:1);
     For I:=1 to NovP Do
         Write(Lst,Track^.VX[I]:8:1);
     Writeln(Lst);
  End
  Else Begin
     Write(PrtFile,Counter:3,Track^.F:9:1,F1:7:0,DF:8:0,FV1:8:0,DF1:9:1);
     For I:=1 to NovP Do
         Write(PrtFile,Track^.VX[I]:8:1);
     Writeln(PrtFile);
  End;
  Track^.FO1:=F1;  Track^.FO2:=FV1;
  Track:=Track^.Next;
  Counter:=Counter + 1;
Until (Track = Nil);
Counter:=Counter - 1;
FA:=FA/Counter;  FB:=FB/Counter;  F1A:=F1A/Counter;
SS2:=0;  SS12:=0;
Track:=FirstARec;
Repeat
  SSA:=Track^.F - FA;
  SS1:=Track^.F - F1A;
  SS12:=SS12 + SS1 * SS1;
  SS2:=SS2 + SSA * SSA;
  Track:=Track^.Next;
```

```
Until (Track = Nil);
Num:=Counter - 1;
If Num <= 0 Then Num:=1;
Writeln('==========================================================');
Write(Counter + 1:3,FA:9:1,FB:7:0,Sqrt(SD2/Num):8:0,F1A:8:0,Sqrt(SDV1/Num):9:1);
For I:=1 to NovT Do
    Write(VS[I]/Counter:8:1);
Writeln;
Writeln('----------------------------------------------------------');
Writeln('|   f1 = a + b*V1');
Writeln('|    a = ',AV1);
Writeln('|    b = ',BV1);
Writeln('|=========================================================');
Writeln('|  f12 = C1 + C2*V1 + C3*V2 + ... + Cn*Vm');
For I:=1 to NumVar + 1 Do
    Writeln('|  C',I:2,' = ',XX[I]);
If PrtOpt = 'Y' Then Begin
   Writeln(Lst,'==========================================================');
   Write(Lst,Counter + 1:3,FA:9:1,FB:7:0,Sqrt(SD2/Num):8:0,F1A:8:0,Sqrt(SDV1/Num):9:1);
   For I:=1 to NovP Do
       Write(Lst,VS[I]/Counter:8:1);
   Writeln(Lst);
   Writeln(Lst,'----------------------------------------------------------');
   Writeln(Lst,'|   f1 = a + b*V1');
   Writeln(Lst,'|    a = ',AV1);
   Writeln(Lst,'|    b = ',BV1);
   Writeln(Lst,'|=========================================================');
   Writeln(Lst,'|  f12 = C1 + C2*V1 + C3*V2 + ... + Cn*Vm');
   For I:=1 to NumVar + 1 Do
       Writeln(Lst,'|  C',I:2,' = ',XX[I]);
End
Else Begin
   Writeln(PrtFile,'==========================================================');
   Write(PrtFile,Counter + 1:3,FA:9:1,FB:7:0,Sqrt(SD2/Num):8:0,F1A:8:0,Sqrt(SDV1/Num):9:1);
   For I:=1 to NovP Do
       Write(PrtFile,VS[I]/Counter:8:1);
   Writeln(PrtFile);
   Writeln(PrtFile,'----------------------------------------------------------');
   Writeln(PrtFile,'|   f1 = a + b*V1');
   Writeln(PrtFile,'|    a = ',AV1);
   Writeln(PrtFile,'|    b = ',BV1);
   Writeln(PrtFile,'|=========================================================');
   Writeln(PrtFile,'|  f12 = C1 + C2*V1 + C3*V2 + ... + Cn*Vm');
   For I:=1 to NumVar + 1 Do
       Writeln(PrtFile,'|  C',I:2,' = ',XX[I]);
End;
```

```
ZC1:=0;  If SS2 > SD2 Then ZC1:=1 - SD2/SS2;  ZC1:=Sqrt(Abs(ZC1));
ZC2:=0;  If SS2 > SDV1 Then ZC2:=1 - SDV1/SS2; ZC2:=Sqrt(Abs(ZC2));
Writeln('   STANDARD DEVIATION OF DATA = ',Sqrt(SS2/Num));
Writeln('   COEF. OF CORRELATION (f12) = ',ZC1);
Writeln('   COEF. OF CORRELATION ( f1) = ',ZC2);
Writeln('   Delta                      = ',ZC1 - ZC2);
If PrtOpt = 'Y' Then Begin
   Writeln(Lst,'   STANDARD DEVIATION OF DATA = ',Sqrt(SS2/Num));
   Writeln(lst,'   COEF. OF CORRELATION (f12) = ',ZC1);
   Writeln(Lst,'   COEF. OF CORRELATION ( f1) = ',ZC2);
   Writeln(Lst,'   Delta                      = ',ZC1 - ZC2);
End
Else Begin
   Writeln(PrtFile,'   STANDARD DEVIATION OF DATA = ',Sqrt(SS2/Num));
   Writeln(PrtFile,'   COEF. OF CORRELATION (f12) = ',ZC1);
   Writeln(PrtFile,'   COEF. OF CORRELATION ( f1) = ',ZC2);
   Writeln(PrtFile,'   Delta                      = ',ZC1 - ZC2);
   Close(PrtFile);
End;
I:=1;
While ((IFileName[I] <> ' ') and (I <=25)) Do I:=I + 1;
I:=I - 1;
FPName:='';
FPName:=IFileName;
I:=Pos('.',FPName);
Delete(FPName,I,25);
FPName:=FPName + '.PLT';
PFileName:=FPName;
Writeln;
Writeln('PLOT FILE ',PFileName);
Write('SAVE DATA [Y/N]: ');
Ans:=' ';
Readln(Trm,Ans);
Ans:=UpCase(Ans);
If Ans = 'Y' Then Begin
   Assign(PFile,PFileName);
   Rewrite(PFile);
   Track:=FirstARec;
   Repeat
      Writeln(PFile,Track^.VX[1]:8:0,Track^.F:10:6,' ',Track^.FO1:9,' ',Track^.FO2:9,' ',0);
      Track:=Track^.Next;
   Until (Track = Nil);
   Close(PFile);
End;
```

```
    Release(HeapTop);
    If PrtOpt <> 'Y' Then Begin
       TempCom:='REGRESSION ANALYSIS REPORT CREATED USING INPUT FILE '
       + IFileName; UpdateCom;
    End;
End.
^Z
```

[BLANK PAGE]

Appendix J.30

RmHead

```
Program RmHead(Input,Output);
{$C-}
```

STEP 1 - Removes Report Headings. The input data file contains report headings, etc. that need to be removed before studies and analysis can be accomplished. The dates are changed to sequential numbers.

Deborah A. Holder

```
TYPE
   HoldName  = array [1..25] of char;
   RegRec = record
                ax,bx,cx,dx,bp,si,di,ds,es,flags: integer;
            end;
   String132 = string[135];
   String255 = string[255];
   Print128  = string[128];
   String50  = string[50];
   String20  = string[20];
   String3   = string[3];
   Char12arr = array [1..12] of char;
   String10  = string[10];
   String6   = string[6];
   DPointer = ^DirRec;
   DirRec = record
              Size: real;
              Hr,Minu: integer;
              Year,Month,Day: integer;
              Next,Prev: DPointer;
              Name: String20;
            end;

   SRec  = record
            SLine: Array[1..70] of char;
           end;

   ImageType = array[1..4096] of char;
   Path = string[64];
```

```
VAR
   K,L,Code,I,ChkLen,Find,J: integer;
   HeapTop: ^Integer;
   BY,NoFiles: integer;
   BYStr: String3;
   Top,Bottom: DPointer;
   FirstRec,LastRec,NewRec: DPointer;
   SFileName,TName,OFileName,IFileName: Path;
   SFile: File of SRec;
   Sqdrec: SRec;
   GFN,InFile,TempFile,OutFile:Text;
   SaveLine,TempLine,Line: String132;
   AnotherTemp: String132;
   PrtStr: Print128;
   HoldStr: String20;
   BX1,BX2,CX1,CX2: String10;
   VX1,VX2,VX3: String10;
   YX1,YX2,YX3: String10;
   CheckStr: String10;
   V1,V2,V3: real;
   V11,V12,V13,V14,V24,V34: real;
   Ans,GFlag,GOpt: Char;
   Field,MaxField: integer;
   UppCase,Default,Quit,ReqFile,ValFile: Boolean;
   Image: ImageType;
   CrtMode: Byte absolute $0040:$0049;
   MonoBuffer:ImageType absolute $B000:$0000;
   ColorBuffer:ImageType absolute $B800:$0000;
   ChkFile,ChkDir,ProgDrvDir,DrvDir: Path;
   TempCom: String255;
   TmpCom1,TmpCom2,TmpCom3,TmpCom4,TmpCom5: Path;
   SqdLine: String[70];
   HlpMsg1,HlpMsg2: String255;


{$IBoxwin.pas}
{$IHelp.pas}
{$IAddDir.pas}
{$IComments.pas}
{$IExists.pas}
{$IDiskSpace.pas}
{$IGetDir.pas}
{$IDirLst.pas}
{$IFileName.pas}
{$IAddFileExt.pas}
{$IReadField.pas}
{$IReadCom.pas}
```

```
{$IDisCom.pas}
{$IUpdateCom.pas}
Begin
    (*  Open Input File and Temp File  *)

    GraphBackGround(Blue);
    TextBackGround(Blue);
    TextColor(Yellow);
    Default:=False;
    ClrScr;
    BoxWin(1,1,80,24,'  STEP 1  ');
    GotoXY(24,10);
    Write('>>> Remove Report Headings <<<');
    GotoXY(1,1);
    GetDir(0,ProgDrvDir);
    Assign(GFN,'GFN_Nail');
    ReSet(GFN);
    ChkDir:=''; ChkFile:=''; DrvDir:='';
    Readln(GFN,DrvDir);
    Readln(GFN,ChkFile);
    Close(GFN);
    ChkDir:=DrvDir;
    If Length(ChkDir) = 2 Then ChkDir:=ChkDir + '\';
    Delay(1000);
    If ((ChkFile <> '**') and (Pos('DAT',ChkFile) > 0)) Then Begin
       IFileName:='';
       IFileName:=ChkFile;
       If DrvDir <> ProgDrvDir Then AddDir(IFileName);
       Default:=True;
    End
    Else Begin
       Mark(HeapTop);
       GetDirectory('????????.DAT');
       ValFile:=False;
       IFileName:='';
       While ValFile = False Do Begin
          IFileName:='                    ';
          ReqFile:=True;
          FileName(IFileName,'  Input   ');
          AddFileExt(IFileName,'DAT');
          If DrvDir <> ProgDrvDir Then AddDir(IFileName);
          ValFile:=Exists(IFileName);
          If ValFile = False Then Begin
             GotoXY(1,12); ClrEOL;
             GotoXY(15,12); Write(^G,'File Does Not Exist !!',IFilename);
             Delay(2000);
```

```
      End;
    End;
    Release(HeapTop);
  End;
  If ((ChkFile <> '**') and (Pos('DAT',ChkFile) > 0)) Then Begin
    OFileName:='';
    OFileName:=ChkFile;
    AddFileExt(OFileName,'DTA');
    ChkFile:=OFileName;
    Assign(GFN,'GFN_Nail');
    Rewrite(GFN);
    Writeln(GFN,DrvDir);
    Writeln(GFN,ChkFile);
    Close(GFN);
    If DrvDir <> ProgDrvDir Then AddDir(OFileName);
    Default:=True;
  End
  Else Begin
    Mark(HeapTop);
    GetDirectory('????????.DTA');
    ValFile:=True;
    OFileName:='';
    Repeat
      OFileName:='                    ';
      ReqFile:=True;
      FileName(OFileName,' Output ');
      AddFileExt(OFileName,'DTA');
      If DrvDir <> ProgDrvDir Then AddDir(OFileName);
      ValFile:=Exists(OFileName);
      If ValFile = True Then Begin
        GotoXY(1,12); ClrEOL;
        GotoXY(1,13); ClrEOL;
        GotoXY(17,12); Write(^G,'File Already Exist !!');
        GotoXY(17,13); Write('Do you wish to overwrite ?: _');
        GotoXY(WhereX -1,WhereY);
        Read(Kbd,Ans);
        Ans:=UpCase(Ans);
        GotoXY(WhereX,WhereY);
        Write(Ans);
        If Ans = 'Y' Then ValFile:=False;
      End;
    Until (ValFile = False);
    Release(HeapTop);
  End;
  TName:='Temp.zzz';
  If DrvDir <> ProgDrvDir Then AddDir(TName);
  ClrScr;
```

```
(* Remove Headings *)
GotoXY(20,10);
Write('STRIPPING HEADING FROM R520 REPORT');
SFileName:='Sqrdqt.sqr';
If DrvDir <> ProgDrvDir Then AddDir(SFileName);
ValFile:=Exists(SFileName);
If ValFile = False Then Begin
   BoxWin(10,10,70,20,' ERROR ');
   GotoXY(3,2);  Write(^G,'Error!!  The R710 Report needs to be processed');
   GotoXY(3,3);  Write('first.   Execute Step 13, Fix R710 Data.');
   Delay(3000);
   Halt;
End;

Assign(InFile,IFileName);
Reset(InFile);
Assign(OutFile,OFileName);
Rewrite(OutFile);
Assign(TempFile,TName);
ReWrite(TempFile);

Assign(SFile,SFileName);
Reset(SFile);
Seek(SFile,0);
Read(SFile,SqdRec);
SqdLine:=SqdRec.SLine;
BYStr:=Copy(SqdLine,7,2);
Val(BYStr,BY,Code);
Close(SFile);
While Not EOF(InFile) Do Begin
   Line:='';
   While Line = '' Do ReadLn(InFile,Line);
   Find:=Pos('R0520',Line);

   If (Find <> 0) Then Begin
      J:=1;
      Line:='';
      While Line = '' Do ReadLn(InFile,Line);
      While ((J <=10) and (Pos('CODE N',Line) = 0)) Do Begin
          Readln(InFile,Line);
          If Line = '' Then Begin
             Readln(InFile,Line);
          End
          Else J:=J + 1;
      End;
      Line:='';
      While Line = '' Do ReadLn(InFile,Line);
```

```
End;

CheckStr:=Copy(Line,25,3);
If (CheckStr <> '   ') Then Begin
   CheckStr:=Copy(Line,1,4);
   If CheckStr <> '    ' Then Begin
      SaveLine:=Line;
      Delete(SaveLine,5,1);
      Insert('&',SaveLine,5);
      TempLine:=Line;
      Delete(Line,1,4);
      Writeln(TempFile,Line);
   End
   Else Begin
      HoldStr:='';
      HoldStr:=Copy(SaveLine,1,20);
      Delete(Line,1,20);
      Insert(HoldStr,Line,1);

      HoldStr:='';
      HoldStr:=Copy(SaveLine,57,13);
      Delete(Line,57,13);
      Insert(HoldStr,Line,57);

      AnotherTemp:=TempLine;
      BX1:=Copy(Line,75,4);
      BX2:=Copy(Line,80,4);

      CX1:=Copy(AnotherTemp,75,4);

      If BX1 < CX1 Then Begin
         Delete(Line,75,4);
         Insert(CX1,Line,75);
      End;

      CX2:=Copy(AnotherTemp,80,4);

      If BX2 < CX2 Then Begin
         Delete(Line,80,4);
         Insert(CX2,Line,80);
      End;

      TempLine:=Line;
      Delete(line,1,4);
      Writeln(TempFile,Line);
```

```
      End;  (* Blanks found in first four spaces of line *)
   End;  (* Blanks in col 25,26,27 found *)
End;   (* While Loop *)

Flush(TempFile);
Close(InFile);

(* Does user wish to exclude G's *)
ClrScr;
If Default = False Then Begin
   GotoXY(20,10);
   Write('Do you wish to EXCLUDE G's [Y,N]: _');
   K:=WhereX - 1; L:=WhereY;
   Repeat
      GOpt:=' ';
      GotoXY(K,L);
      Read(Kbd,GOpt);
      If Ord(GOpt) = 27 Then Begin
         If KeyPressed then Read(Kbd,GOpt);
         If Ord(GOpt) = 59 Then
           Help('Enter  Y  if you wish to exclude G's from the output file.',
           'Enter  N  if you wish to include the G's.')
      End
      Else Begin
         GOpt:=UpCase(GOpt);
         GotoXY(K,L); Write(GOpt);
         If ((GOpt <> 'Y') and (GOpt <> 'N')) Then Begin
            GotoXY(10,15); Write(^G,'Invalid Response');
         End;
      End;
   Until((GOpt = 'Y') or (GOpt = 'N'));
End
Else GOpt:='Y';

Reset(TempFile);

ClrScr;
GotoXY(30,10);
Write('PROCESSING FILE');

While Not EOF(TempFile) Do Begin
    Readln(TempFile,Line);
    GFlag:=' ';
    If GOpt = 'Y' Then
       GFlag:=Copy(Line,126,1);
    If GFlag <> 'G' Then Begin
```

```
        VX1:=Copy(Line,57,5); Val(VX1,V1,Code);
        VX2:=Copy(Line,72,3); Val(VX2,V2,Code);
        VX3:=Copy(Line,77,3); Val(VX3,V3,Code);
        V11:=Int(V1 * 0.001);
        V12:=V1 - V11 * 1000;
        V13:=(V11 - BY) * 365;
        V14:=V13 + V12;
        V24:=V2 + V13;
        V34:=V3 + V13;

        Str(V14:5:0,YX1);
        Str(V24:5:0,YX2);
        Str(V34:5:0,YX3);

        ChkLen:=Length(YX1);
        If ChkLen < 5 Then
           For I:=1 to 5 - ChkLen Do Begin
               Insert(' ',YX1,1);
               Insert(' ',YX2,1);
               Insert(' ',YX3,1);
           End;
        If V1 > 0 Then Begin
               Delete(Line,57,5);
               Insert(YX1,Line,57);
        End;
        If V2 > 0 Then Begin
               Delete(Line,71,5);
               Insert(YX2,Line,71);
        End;
        If V3 > 0 Then Begin
               Delete(Line,76,5);
               Insert(YX3,Line,76);
        End;

        If Length(Line) < 128 Then
           For I:=Length(Line) + 1 to 128 Do Line:=Line + ' ';
        Writeln(OutFile,Line);
      End;
    End;

    Close(TempFile);
    Erase(TempFile);
    Flush(OutFile);
    Close(OutFile);
    TempCom:='REMOVE REPORT HEADING.' + '  INPUT FILE => ' + IFilename;
    UpDateCom;

End.
^Z
```

Appendix J.31

SortStep

```
PROGRAM          SORTSTEP;

  TYPE
     PHRASES   = STRING[140];
     CHAR12ARR = ARRAY[1..12] OF CHAR;
     HOLDNAME  = ARRAY[1..25] OF CHAR;
     STRING3   = STRING[3];
     STRING10  = STRING[10];
     STRING20  = STRING[20];
     STRING50  = STRING[50];
     STRING255 = STRING[255];
     PATH      = STRING[64];
     IMAGETYPE = ARRAY[1..4096] OF CHAR;
     DPOINTER  = ^DIRREC;

     DIRREC    = RECORD
            SIZE     : REAL;
            HR       : INTEGER;
            MINU     : INTEGER;
            YEAR     : INTEGER;
            MONTH    : INTEGER;
            DAY      : INTEGER;
            NEXT     : DPOINTER;
            PREV     : DPOINTER;
            NAME     : STRING20;
     END;

     REGREC    = RECORD
            AX       : INTEGER;
            BX       : INTEGER;
            CX       : INTEGER;
            DX       : INTEGER;
            BP       : INTEGER;
            SI       : INTEGER;
            DI       : INTEGER;
            DS       : INTEGER;
            ES       : INTEGER;
            FLAGS    : INTEGER;
     END;

     INDATA    = RECORD
            DATAREC  : STRING[140];
     END;
```

```
VAR
    EXCSTR       : PHRASES;
    INCSTR       : PHRASES;
    SORTED       : TEXT;
    UNSORT       : TEXT;
    GFN          : TEXT;
    IMAGE        : IMAGETYPE;
    COLORBUFFER : IMAGETYPE ABSOLUTE $B800:$0000;
    MONOBUFFER  : IMAGETYPE ABSOLUTE $B000:$0000;
    CRTMODE      : BYTE ABSOLUTE $0040:$0049;
    INREC        : INDATA;
    NEWBIG       : INDATA;
    OLDBIG       : INDATA;
    WORKFILE     : FILE OF INDATA;
    IFILENAME    : PATH;
    OFILENAME    : PATH;
    CHKDIR,
    PROGDRVDIR,
    DRVDIR       : PATH;
    BOTTOM       : DPOINTER;
    FIRSTREC     : DPOINTER;
    LASTREC      : DPOINTER;
    NEWREC       : DPOINTER;
    TOP          : DPOINTER;
    KEYVAL       : PHRASES;
    LINEIN       : PHRASES;
    NCLUDE       : PHRASES;
    OLDVAL       : PHRASES;
    XCLUDE       : PHRASES;
    ANS          : CHAR;
    REQFILE      : BOOLEAN;
    VALFILE      : BOOLEAN;
    HEAPTOP      : ^INTEGER;
    INCBEG       : INTEGER;
    EXCBEG       : INTEGER;
    SKBEG        : INTEGER;
    SKEND        : INTEGER;
    BOTTOMLOOP  : INTEGER;
    CHANGES      : INTEGER;
    EXCCOL       : INTEGER;
    I            : INTEGER;
    INCCOL       : INTEGER;
    J            : INTEGER;
    LNEX         : INTEGER;
    LNIN         : INTEGER;
    LNOUT        : INTEGER;
    LNSK         : INTEGER;
```

```
    NEWTOP      : INTEGER;
    NFLAG       : INTEGER;
    NOFILES     : INTEGER;
    NOSORTS     : INTEGER;
    NOWRIT      : INTEGER;
    SKLAST      : INTEGER;
    SKSTART     : INTEGER;
    TOPLOOP     : INTEGER;
    XFLAG       : INTEGER;
    INTFILE     : PATH;
    TSIN        : STRING[25];

{$IBOXWIN.PAS}
{$IHELP.PAS}
{$IADDDIR.PAS}
{$ICOMMENTS.PAS}
{$IEXISTS.PAS}
{$IDISKSPACE.PAS}
{$IGETDIR.PAS}
{$IDIRLST.PAS}
{$IFILENAME.PAS}
{$IADDFILEEXT.PAS}

BEGIN
    GRAPHBACKGROUND(BLUE);
    TEXTBACKGROUND(BLUE);
    TEXTCOLOR(YELLOW);
    CLRSCR;
    BOXWIN(1,1,80,24,'  STEP 20 ');
    GETDIR(0,PROGDRVDIR);
    GOTOXY(30,10);
    WRITE ('>>> SORT FILE <<<');
    GOTOXY(1,1);
    ASSIGN(GFN,'GFN_NAIL');
    RESET(GFN);
    READLN(GFN,DRVDIR);
    CLOSE(GFN);
    CHKDIR:=DRVDIR;
    IF LENGTH(CHKDIR) = 2 THEN CHKDIR:=CHKDIR + '\';
    SKBEG:=0;
    SKEND:=0;
    INCBEG:=0;
    EXCBEG:=0;
    EXCSTR:='';
    INCSTR:='';
    DELAY(1000);
```

```
MARK(HEAPTOP);
GETDIRECTORY('????????.D??');
VALFILE:=FALSE;
WHILE VALFILE = FALSE DO BEGIN
    IFILENAME:='                    ';
    REQFILE:=TRUE;
    FILENAME(IFILENAME,' Input  ');

    IF PROGDRVDIR <> DRVDIR THEN ADDDIR(IFILENAME);

    VALFILE:=EXISTS(IFILENAME);
    IF VALFILE = FALSE THEN BEGIN
        GOTOXY(1,12);
        CLREOL;
        GOTOXY(15,12);
        WRITE(^G,'File Does Not Exist !!');
    END;
END;

VALFILE:=TRUE;
REPEAT
    OFILENAME:='                    ';
    REQFILE:=TRUE;
    FILENAME(OFILENAME,' Output  ');
    IF PROGDRVDIR <> DRVDIR THEN ADDDIR(OFILENAME);
    VALFILE:=EXISTS(OFILENAME);
    IF VALFILE = TRUE THEN BEGIN
        GOTOXY(1,12);
        CLREOL;
        GOTOXY(1,13);
        CLREOL;
        GOTOXY(17,12);
        WRITE(^G,'File Already Exists !!');
        GOTOXY(17,13);
        WRITE('Do you wish to overwrite ?: _');
        GOTOXY(WHEREX -1,WHEREY);
        READ(KBD,ANS);
        ANS:=UPCASE(ANS);
        GOTOXY(WHEREX,WHEREY);
        WRITE(ANS);
        IF ANS = 'Y' THEN VALFILE:=FALSE;
    END;
UNTIL (VALFILE = FALSE);
RELEASE(HEAPTOP);
```

```
CLRSCR;
GOTOXY(20,10);
CLREOL;
WRITE ('BEGIN SORT IN COLUMN: ');
READLN (trm,SKBEG);
GOTOXY(20,10);
CLREOL;
WRITE ('END SORT IN COLUMN: ');
READLN (trm,SKEND);
GOTOXY(20,10);
CLREOL;
WRITE ('INCLUDE LINES CONTAINING STRING: ');
READLN (trm,INCSTR);
IF INCSTR <> '' THEN BEGIN
  GOTOXY(20,12);
  CLREOL;
  WRITE ('WHICH START IN COLUMN: ');
  READLN (trm,INCBEG);
END;
CLRSCR;
GOTOXY(20,10);
CLREOL;
WRITE ('EXCLUDE LINES CONTAINING STRING: ');
READLN (trm,EXCSTR);
IF EXCSTR <> '' THEN BEGIN
  GOTOXY(20,12);
  CLREOL;
  WRITE ('WHICH START IN COLUMN: ');
  READLN (trm,EXCBEG);
END;

CLRSCR;
GOTOXY(27,11);
WRITE ('>>> SORTING FILE <<<');
INTFILE:='WORK.XXX';
IF PROGDRVDIR <> DRVDIR THEN ADDDIR(INTFILE);
SKSTART:=SKBEG;
SKLAST:=SKEND;
INCCOL:=INCBEG;
EXCCOL:=EXCBEG;
XCLUDE:=EXCSTR;
NCLUDE:=INCSTR;
LNIN:=LENGTH(NCLUDE);
LNEX:=LENGTH(XCLUDE);
LNSK:=(SKLAST-SKSTART)+1;
ASSIGN (UNSORT,IFILENAME);
RESET (UNSORT);
```

```
ASSIGN (WORKFILE,INTFILE);
REWRITE (WORKFILE);
NOWRIT:=0;
WHILE NOT EOF(UNSORT) DO BEGIN
    READLN(UNSORT,LINEIN);
    INREC.DATAREC:=LINEIN;
    NFLAG:=0;
    XFLAG:=0;
    IF LNIN > 0 THEN BEGIN
        KEYVAL:=COPY(INREC.DATAREC,INCCOL,LNIN);
        IF KEYVAL = NCLUDE THEN
            NFLAG:=1
        ELSE
            NFLAG:=0;
    END
    ELSE
        NFLAG:=1;
    IF LNEX > 0 THEN BEGIN
        KEYVAL:=COPY(INREC.DATAREC,EXCCOL,LNEX);
        IF KEYVAL <> XCLUDE THEN
            XFLAG:=1
        ELSE
            XFLAG:=0;
    END
    ELSE
        XFLAG:=1;
    IF (NFLAG = 1) AND (XFLAG = 1) THEN BEGIN
        NOWRIT:=NOWRIT+1;
        WRITE(WORKFILE,INREC);
    END;

END;
CLOSE(UNSORT);
BOTTOMLOOP:=1;
TOPLOOP:=NOWRIT;
NEWTOP:=TOPLOOP;
CHANGES:=1;
OLDVAL:=COPY(INREC.DATAREC,SKSTART,LNSK);
WHILE CHANGES > 0 DO BEGIN
    CHANGES:=0;
    FOR I:=BOTTOMLOOP TO TOPLOOP DO BEGIN
        SEEK(WORKFILE,I-1);
        READ(WORKFILE,INREC);
        KEYVAL:=COPY(INREC.DATAREC,SKSTART,LNSK);
        IF KEYVAL > OLDVAL THEN BEGIN
            OLDVAL:=KEYVAL;
            NEWTOP:=I;
```

```
                CHANGES:=CHANGES+1;
            END;
        END;
        IF CHANGES > 0 THEN BEGIN
            SEEK(WORKFILE,NEWTOP-1);
            READ(WORKFILE,NEWBIG);
            SEEK(WORKFILE,TOPLOOP-1);
            READ(WORKFILE,OLDBIG);
            SEEK(WORKFILE,TOPLOOP-1);
            WRITE(WORKFILE,NEWBIG);
            SEEK(WORKFILE,NEWTOP-1);
            WRITE(WORKFILE,OLDBIG);
            OLDVAL:='';
            TOPLOOP:=TOPLOOP-1;
        END;
    END;
    ASSIGN(SORTED,OFILENAME);
    REWRITE(SORTED);
    FOR I:=1 TO NOWRIT DO BEGIN
        SEEK(WORKFILE,I-1);
        READ(WORKFILE,INREC);
        WRITELN(SORTED,INREC.DATAREC);
    END;
    CLOSE(SORTED);
    CLOSE(WORKFILE);

END.
```

^Z

[BLANK PAGE]

Appendix J.32

Time

```
Function Time: DateTimeType;


Type
  RegPack = record
              AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags: Integer;
            end;

Var
  Regs:   RegPack;
  Hour,Min,Sec,Temp: DateTimeType;

Begin

  With Regs do Begin
     AX:=$2C00;
     MsDos(Regs);

     Str(hi(CX):2,Hour);
     Str(lo(CX):2,Min);
     Str(hi(DX):2,Sec);
  End;

  Temp:=Hour + ':' + Min + ':' + Sec;
  For I:=1 to Length(Temp) Do If Temp[I] = ' ' Then Temp[I]:='0';
  Time:=Temp;

End;
^Z
```

[BLANK PAGE]

Appendix J.33

UpDateCom

```
Procedure UpDateCom;

VAR
   DirName: Char12arr;
   J,I: integer;
   ComFile,Temp: text;
   CName,TName: Path;
   Found: Boolean;
   DirLine,ComLine: String255;
   Check: Path;

(*==============================================================*)
  Procedure CommentScr;

  Begin  (* Procedure CommentScr *)

    ClrScr;
    GotoXY(31,2); Write('UPDATE COMMENT FILE');
    GotoXY(5,5);  Write('File Name: ',FirstRec^.Name);
    GotoXY(5,6);  Write('File Size: ',FirstRec^.Size:7:0,' bytes');
    GotoXY(5,8);  Write('Date: ',FirstRec^.Month,'/');
    If FirstRec^.Day < 10 Then Write('0');
    Write(FirstRec^.Day,'/',FirstRec^.Year);
    GotoXY(5,9);  Write('Time: ',FirstRec^.Hr,':');
    If FirstRec^.Minu < 10 Then Write('0');
    Write(FirstRec^.Minu);
    GotoXY(5,11); Write('Comments: ________________________________________');
    GotoXY(5,12); Write('          ________________________________________');
    GotoXY(5,13); Write('          ________________________________________');
    GotoXY(5,14); Write('          ________________________________________');
    GotoXY(5,15); Write('          ________________________________________');
    TextBackGround(LightGray);
    TextColor(Black);
    GotoXY(5,22); Write('  F1:Help                          Esc Key: Exit to NAIL  ');
    TextBackGround(Blue);
    TextColor(Yellow);
  End;  (* Procedure CommentScr *)
(*==============================================================*)

Begin  (* Procedure UpDateCom *)

  I:=Length(OFileName);
  While OFileName[I] = ' ' Do Begin
```

```
   Delete(OFileName,I,1);
   I:=I - 1;
End;
For J:=1 to 12 Do DirName[J]:=' ';
I:=Length(OFileName);  J:=12;
While ((OFileName[I] <> '\') and (OFileName[I] <> ':') and (J > 0) and (I > 0)) Do Begin
    DirName[J]:=OFileName[I];  J:=J - 1;  I:=I - 1;
End;
While DirName[1] = ' ' Do Begin
   For I:=1 to 11 Do
       DirName[I]:=DirName[I + 1];
   DirName[12]:=' ';
End;

Mark(HeapTop);
GetDirectory(DirName);
If Default = False Then CommentScr;
Check:='';
Check:=FirstRec^.Name;
Release(HeapTop);
DisplayCom;
If Default = False Then ReadCom;
BuildCom;
ClrScr;
CName:='DataList.cmt';
TName:='Temp.zzz';
If DrvDir <> ProgDrvDir Then Begin
   AddDir(CName);
   AddDir(TName);
End;
Assign(ComFile,CName);
Reset(ComFile);
Assign(Temp,TName);
ReWrite(Temp);
Readln(ComFile,ComLine);
Writeln(Temp,ComLine);
Found:=False;
While ((NOT EOF(ComFile)) and (Found = False)) Do Begin
   Readln(ComFile,DirLine);
   ComLine:='';
   If Not EOF(ComFile) Then Readln(ComFile,ComLine);
   For I:=1 to Length(DirLine) Do DirLine[I]:=UpCase(DirLine[I]);
   If (Pos(Check,DirLine)) <> 0 Then Found:=True
   Else Begin
       Writeln(Temp,DirLine);
       Writeln(Temp,ComLine);
```

```
      End;
    End;
    Writeln(Temp,Check);
    Writeln(Temp,TempCom);
    While NOT EOF(ComFile) Do Begin
      Readln(ComFile,DirLine);
      Writeln(Temp,DirLine);
    End;
    Close(ComFile);
    Close(Temp);
    Erase(ComFile);
    ReName(Temp,CName);

  End;  (* Procedure UpDateCom *)^Z
```

[BLANK PAGE]

Appendix J.34

ValDir

```
Function ValDir(Dir:Path):Boolean;
=================================
 Determine if Valid Drive/Directory
=================================

Begin  (* Function ValDir *)

  {$I-}
  ChDir(Dir);
  {$I+}
  ValDir:=(IOresult = 0);
  ChDir(ProgDrvDir);

End;  (* Function ValDir *)
^Z
```

[BLANK PAGE]

APPENDIX J.35

HPDRAW.BAS

```
10 CLS
20 T$=TIME$:T=VAL(LEFT$(T$,2)):FOR I=1 TO T: X=RND:NEXT I
30 GOSUB 2680
40 DIM X(1500),Y(1500,3),I(1500)
50 MAX=1500: VTY=10: VTX=10
60 SCAX=1 :SCAY=1: SCAP=1: SCYF=1: SCXF=1: CHX=1: CHY=1
70 YX=9700: XX=15700: YM=1000: XM=1500
80 INPUT "Change scale factors for x,y, and/or pen vel. (Y/N) ";SF$
90 SF$= LEFT$(SF$,1)
100 IF SF$<>"Y" AND SF$<>"y" THEN 270
110 INPUT "Input X scale factor (.01-.99) or RETURN ";SF$
120 IF SF$<>"" THEN SCAX=VAL(SF$)
130 INPUT "Input Y scale factor (.01-.99) or RETURN ";SF$
140 IF SF$<>"" THEN SCAY=VAL(SF$)
150 INPUT "Input P scale factor (.01-.99) or RETURN ";SF$
160 IF SF$<>"" THEN SCAP=VAL(SF$)
170 INPUT "Rotate coordinate system    90 or RETURN ";SF$
180 IF SF$<>"" THEN RO=90
190 INPUT "Input X MIN   factor (.01-9.9) or RETURN ";SF$
200 IF SF$<>"" THEN SCXF=VAL(SF$)
210 INPUT "Input Y MIN   factor (.01-9.9) or RETURN ";SF$
220 IF SF$<>"" THEN SCYF=VAL(SF$)
230 INPUT "Input Y char. factor (.01-9.9) or RETURN ";SF$
240 IF SF$<>"" THEN CHY=VAL(SF$)
250 INPUT "Input X char. factor (.01-9.9) or RETURN ";SF$
260 IF SF$<>"" THEN CHX=VAL(SF$)
270 PENSPEED=38.1*SCAP
280 YX=YM+(YX-YM)*SCAY: YM=YM*SCYF: CH=.375*SCAY*CHY:
CW=.285*SCAX*CHX
290 XX=XM+(XX-XM)*SCAX: XM=XM*SCXF
300 CLS
310 VDY=1/VTY: VDX=1/VTX
320 PI=3.1415927#
330 NP=2
340 GOSUB 1860
350 PN=1
351 GOSUB 2230
360 GOSUB 2190
370 GOSUB 1950
```

```
390 ON ERR GOTO 1840
400 INPUT "Input plot file = ";FI$
410 OPEN FI$ FOR OUTPUT AS 1
1820 TYPE=0
1830 GOSUB 2070
1840 PRINT #NP,"sp0;"
1850 CLOSE: END
1860 'Open # NP as plot output
1870 '=====================
1880 OPEN "com1:9600,s,7,1,RS,CS65535,DS,CD" AS # NP
1890 PRINT # NP,"IN;"
1900 RETURN
1910 'Get pen # PN
1920 '=====================
1930 PRINT # NP, "SP ";PN;";"
1940 RETURN
1950 'Set pen velocity to PENSPEED
1960 '=====================
1970 PRINT # NP,"VS";PENSPEED;";"
1980 RETURN
1990 'Move pen to X,Y
2000 '=====================
2010 PRINT # NP,"PA";X;Y;";"
2020 RETURN
2030 'Pen down
2040 '=====================
2050 PRINT # NP,"PD;"
2060 RETURN
2070 'Pen UP
2080 '=====================
2090 PRINT # NP,"PU;"
2100 RETURN
2110 'Write TIT$ at X,Y of set CL
2120 '=====================
2130 GOSUB 2070
2140 GOSUB 1990
2150 PRINT #NP,"CP";CL;CD;";"
2160 CD=0:CL=0
2170 PRINT # NP,"LB";TIT$;CHR$(3)
2180 RETURN
2190 'Set character height (CH) and width (CW)
2200 '=====================
2210 PRINT # NP,"SI";CW;CH;";"
2220 RETURN
2230 'Rotate coordinate system RO degrees (0 or 90)
```

```
2240 '=====================
2250 PRINT # NP,"RO";RO;";"
2260 RETURN
2270 'Set character run (CX) and rise (CY)
2280 '=====================
2290 PRINT # NP,"DI";CX;CY;";"
2300 RETURN
2310 'Plot circle at X,Y with radius RC and cord angle CC
2320 '=====================
2330 GOSUB 1990
2340 PRINT # NP,"CI";RC;CC;";"
2350 RETURN
2360 'Plot box at X,Y with height BH and width BW
2370 '=====================
2380 GOSUB 2070
2390 GOSUB 1990
2400 GOSUB 2030
2410 PRINT # NP,"EA";X+BW;Y+BH;";"
2420 GOSUB 2070
2430 RETURN
2440 'Fill box at X,Y with height BH and width BW
2450 '=====================
2460 GOSUB 1990
2470 PRINT # NP,"RA";X+BW;Y+BH;";"
2480 RETURN
2490 'Fill TYPE =1 & 2 (Solid), 3 (Lines), 4,5 & 6 (Cross-hatch)
2500 '=====================
2510 DD=0:BA=0:IF TYPE=3 THEN DD=100
2520 IF TYPE=4 THEN DD= 50: BA=45
2530 IF TYPE=5 THEN DD= 50: BA=90
2540 IF TYPE=6 THEN DD= 50: BA=135
2550 PRINT # NP,"FT";TYPE;DD;BA;";"
2560 RETURN
2570 'Select new pen
2580 '=====================
2590 PN=(PN MOD 6) +1
2600 GOSUB 1910
2610 RETURN
2620 'Draw box at X,Y to X1,Y1
2630 '=====================
2640 GOSUB 1990:GOSUB 2040
2650 PRINT # NP,"PA";X1;Y;X1;Y1;X;Y1;X;Y;";"
2660 GOSUB 2070
2670 RETURN
2680 'radium variation
```

```
2690 F$=TIME$:G$=RIGHT$(F$,2)+LEFT$(F$,2)
2700 T=VAL(G$) MOD 501
2710 LOCATE 10,50:PRINT F$;T
2720 FOR K=1 TO T:L=RND
2730 NEXT K:SA=INT(RND*3.5+12):SB=INT(RND*7.9):SC=(SB+RND*7) MOD 7
2740 COLOR SA,SB,SC

2750 RETURN
2760 'Write NUMB at X,Y of set CL
2770 '======================
2780 HR=0
2790 TIT$=STR$(NUMB)
2800 LN=LEN(TIT$)
2810 IF LN<7 THEN 2900
2820 IF HR=1 THEN 2840
2830 NUMB=INT(NUMB*1000+.499)/1000:HR=1:GOTO 2790
2840 '
2850 AA=NUMB-INT(NUMB): BB=1-AA
2860 IF AA<1 THEN 2900
2870 IF 10*AA >1 THEN 2890
2880 BB=BB/10
2890 ZZ=INT(NUMB*1000+BB*1000)/1000:NUMB=ZZ:TIT$=STR$(ZZ)
2900 IF LN<6 THEN TIT$=STRING$(6-LN," ")+TIT$
2910 GOSUB 2110
2920 RETURN
2930 'Do a bar form 0 to Y that is DEL-X wide
2940 '======================
2950 IF I<>MAX THEN DEL = BX*(X(I+1)-X(I))
2960 IF I=MAX THEN DEL = BX*(X(I)-X(I-1))
2970 BH=AY-Y: BW=-DEL
2980 X=X+DEL*.5
2990 GOSUB 2360
3000 IF TYPE<2 THEN 3030
3010 GOSUB 2490
3020 GOSUB 2440
3030 RETURN
3040 'Plot point of type PO
3050 '======================
3060 PSZ=TICY:IF TICX<PSZ THEN PSZ=TICX
3070 IF PO<3 THEN 3140
3080 X=X+2*TICX: GOSUB 1990
3090 GOSUB 2030
3100 X=X-3*TICX:Y=Y-TICY:GOSUB 1990
3110 Y=Y+2*TICY:GOSUB 1990
3120 X=X+3*TICX:Y=Y-TICY:GOSUB 1990
```

```
3130 GOTO 3150
3140 GOSUB 1990: CC=120-30*PO: RC=PSZ: GOSUB 2030:GOSUB 2310
3150 GOSUB 2070: RETURN
^Z
```

[BLANK PAGE]

APPENDIX J.36

HPPLOT.BAS

```
  2 DIM DIR$(512):MAX=512:GOSUB 50720:COLOR 0,7:CLS
 10 CLS
 20 T$=TIME$:T=VAL(LEFT$(T$,2)):FOR I=1 TO T: X=RND:NEXT I
 30 GOSUB 2680
 40 DIM X(1500),Y(1500,3),I(1500)
 50 MAX=1500: VTY=10: VTX=10
 60 SCAX=1 :SCAY=1: SCAP=1: SCYF=1: SCXF=1: CHX=1: CHY=1
 70 YX=9700: XX=15700: YM=1000: XM=1500
 80 INPUT "Change scale factors for x,y, and/or pen vel. (Y/N) ";SF$
 90 SF$= LEFT$(SF$,1)
100 IF SF$<>"Y" AND SF$<>"y" THEN 270
110 INPUT "Input X scale factor (.01-.99) or RETURN ";SF$
120 IF SF$<>"" THEN SCAX=VAL(SF$)
130 INPUT "Input Y scale factor (.01-.99) or RETURN ";SF$
140 IF SF$<>"" THEN SCAY=VAL(SF$)
150 INPUT "Input P scale factor (.01-.99) or RETURN ";SF$
160 IF SF$<>"" THEN SCAP=VAL(SF$)
170 INPUT "Rotate coordinate system    90 or RETURN ";SF$
180 IF SF$<>"" THEN RO=90
190 INPUT "Input X MIN   factor (.01-9.9) or RETURN ";SF$
200 IF SF$<>"" THEN SCXF=VAL(SF$)
210 INPUT "Input Y MIN   factor (.01-9.9) or RETURN ";SF$
220 IF SF$<>"" THEN SCYF=VAL(SF$)
230 INPUT "Input Y char. factor (.01-9.9) or RETURN ";SF$
240 IF SF$<>"" THEN CHY=VAL(SF$)
250 INPUT "Input X char. factor (.01-9.9) or RETURN ";SF$
260 IF SF$<>"" THEN CHX=VAL(SF$)
270 PENSPEED=38.1*SCAP
280 YX=YM+(YX-YM)*SCAY: YM=YM*SCYF:CH=.375*SCAY*CHY: CW=.285*SCAX*CHX
290 XX=XM+(XX-XM)*SCAX: XM=XM*SCXF
300 CLS
310 VDY=1/VTY: VDX=1/VTX
320 PI=3.1415927#
330 NP=2
340 GOSUB 1860
350 PN=1
360 GOSUB 2230
370 GOSUB 2190
380 GOSUB 1950
390 ON ERR GOTO 1840
```

```
400 INPUT "Input plot file = ";FI$
401 IF FI$<>"" THEN 410
402 GOSUB 50000
403 FI$=DEFILE$
410 OPEN FI$ FOR INPUT AS 1
420 INPUT #1,T1$
430 INPUT #1,T2$
440 INPUT #1,T3$
445 input #1,dumb$
450 XL=999999!:XH=-XL:Y1L=XL:Y1H=XH:Y2L=Y1L:Y2H=Y1H:Y3L=Y2L:Y3H=Y2H
460 FOR I=1 TO MAX
470 IF EOF(1) THEN 580
480 INPUT #1,X(I),Y(I,1),Y(I,2),Y(I,3)
490 IF X(I)<XL THEN XL=X(I)
500 IF X(I)>XH THEN XH=X(I)
510 IF Y(I,1)>Y1H THEN Y1H=Y(I,1)
520 IF Y(I,1)<Y1L THEN Y1L=Y(I,1)
530 IF Y(I,2)>Y2H THEN Y2H=Y(I,2)
540 IF Y(I,2)<Y2L THEN Y2L=Y(I,2)
550 IF Y(I,3)>Y3H THEN Y3H=Y(I,3)
560 IF Y(I,3)<Y3L THEN Y3L=Y(I,3)
570 NEXT I
580 MAX=I-1
590 DX=XH-XL
600 DY3=Y3H-Y3L
610 DY2=Y2H-Y2L
620 YL=Y1L:IF Y2L<YL THEN YL=Y2L
630 IF Y3L<YL THEN YL=Y3L
640 YH=Y1H:IF Y2H>YH THEN YH=Y2H
650 IF Y3H>YH THEN YH=Y3H
660 DY=YH-YL
670 PRINT "X-HIGH = ";XH
680 PRINT "X-LOW  = ";XL
690 PRINT "--------------------"
700 PRINT "X-DEL  = ";DX
710 PRINT "                    "
720 PRINT "Y-HIGH = ";YH
730 PRINT "Y-LOW  = ";YL
740 PRINT "--------------------"
750 PRINT "Y-DEL  = ";DY
760 PRINT "                    "
770 INPUT " Input X-max = ";XMAX
780 INPUT " Input X-min = ";XMIN
790 INPUT " Input Y-max = ";YMAX
800 INPUT " Input Y-min = ";YMIN
```

```
 810 INPUT " Input X-div = ";VTX$
 820 INPUT " Input Y-div = ";VTY$
 830 IF VTX$<>"" THEN VTX=VAL(VTX$)
 840 IF VTY$<>"" THEN VTY=VAL(VTY$)
 850 HERE=CSRLIN
 860 VDY=1/VTY: VDX=1/VTX
 870 LOCATE 2,50:PRINT "---------------------"
 880 LOCATE 3,50:PRINT "| Code Output type |"
 890 LOCATE 4,50:PRINT "|----- ------------|"
 900 LOCATE 5,50:PRINT "|  0   No    Plot  |"
 910 LOCATE 6,50:PRINT "|  1   Line  Plot  |"
 920 LOCATE 7,50:PRINT "|  2   Bar   Plot  |"
 930 LOCATE 8,50:PRINT "|  3   Symb. Plot  |"
 940 LOCATE 9,50:PRINT "====================="
 950 LOCATE 11,45:INPUT "Input Code for line # 1 = ";C1$
 960 LOCATE 12,45:INPUT "Input Code for line # 2 = ";C2$
 970 LOCATE 13,45:INPUT "Input Code for line # 3 = ";C3$
 980 LOCATE 15,45:INPUT "Color Code  lines (Y/N)   ";YN$
 990 YN=0: YN$= LEFT$(YN$,1): IF YN$="y" OR YN$="Y" THEN YN=1
1000 C1=VAL(C1$)
1010 C2=VAL(C2$)
1020 C3=VAL(C3$)
1030 IF C1>1 THEN O1=O1+1
1040 IF C2>1 THEN O2=O1+1
1050 IF C3>1 THEN O3=O2+1
1060 LOCATE HERE,1
1070 BX=(XX-XM)/(XMAX-XMIN)
1080 AX=XM-BX*XMIN
1090 BY=(YX-YM)/(YMAX-YMIN)
1100 AY=YM-BY*YMIN
1110 GOSUB 1910
1120 X=XM:Y=YM:X1=XX*1.02:Y1=YX:GOSUB 2620
1130 L1=LEN(T1$):CL=-L1*.5
1140 L2=LEN(T2$):L3=LEN(T3$)
1150 T1=(XX-XM)*.5+XM
1160 X=T1:Y=YX:TIT$=T1$:CD=1:GOSUB 2110
1170 Y=YM:CL=-L3*.5:CD=-2.5:TIT$=T3$:GOSUB 2110
1180 CX=0:CY=1:GOSUB 2270
1190 CD= 4:X=XM:Y=(YX-YM)*.5+YM:CL=-L2*.5:TIT$=T2$:GOSUB 2110
1200 CX=1:CY=0:GOSUB 2270
1210 DY=YMAX-YMIN: DX= XMAX-XMIN
1220 YS=YMIN: XS=XMIN
1230 TIY=(YX-YM)
1240 TIX=(XX-XM)
1250 TICY=TIY*.01
```

```
1260 TICX=TIX*.01
1270 FOR I=1 TO VTY +1
1280 X=XM
1290 NUMB=YS + DY*(I-1)*VDY
1300 Y=YM+TIY*VDY*(I-1)
1310 GOSUB 1990:GOSUB 2030
1320 X=X-TICX:GOSUB 1990:GOSUB 2070
1330 CL=-6: GOSUB 2760
1340 NEXT I
1350 FOR I=1 TO VTX + 1
1360 Y=YM
1370 NUMB=XS + DX*(I-1)*VDX
1380 X=XM+TIX*VDX*(I-1)
1390 GOSUB 1990:GOSUB 2030
1400 Y=Y-TICY:GOSUB 1990:GOSUB 2070
1410 CD=-1:CL=-5: GOSUB 2760
1420 NEXT I
1430 IF C1=0 THEN 1550
1440 X=X(1)*BX+AX: Y=Y(1,1)*BY+AY:GOSUB 1990
1450 GOSUB 2030
1460 IF YN=1 THEN PN=PN+1:GOSUB 2570
1470 FOR I=1 TO MAX
1480 X=X(I)*BX+AX: Y=Y(I,1)*BY+AY
1490 IF X>XX THEN 1530
1500 IF C1=1 THEN GOSUB 1990
1510 IF C1=2 THEN GOSUB 2930
1520 IF C1=3 THEN PO=0:GOSUB 3040
1530 NEXT I
1540 TYPE=0
1550 IF C2=0 THEN 1690
1560 GOSUB 2070
1570 X=X(MAX)*BX+AX: Y=Y(MAX,2)*BY+AY:GOSUB 1990
1580 GOSUB 2030
1590 IF O2>1 THEN TYPE=(O2 MOD 6)+1
1600 PN=1:IF YN=1 THEN PN=3:GOSUB 2570
1610 FOR I=MAX TO 1 STEP -1
1620 X=X(I)*BX+AX: Y=Y(I,2)*BY+AY
1630 IF X>XX THEN 1670
1640 IF C2=1 THEN GOSUB 1990
1650 IF C2=2 THEN GOSUB 2930
1660 IF C2=3 THEN PO=1:GOSUB 3040
1670 NEXT I
1680 TYPE=0
1690 IF C3=0 THEN 1830
1700  GOSUB 2070
```

```
1710 X=X(1)*BX+AX: Y=Y(1,3)*BY+AY:GOSUB 1990
1720 GOSUB 2030
1730 IF O3>1 THEN TYPE=(O3 MOD 6)+1
1740 PN=1:IF YN=1 THEN PN=3:GOSUB 2570
1750 FOR I=1 TO MAX
1760 X=X(I)*BX+AX: Y=Y(I,3)*BY+AY
1770 IF X>XX THEN 1810
1780 IF C3=1 THEN GOSUB 1990
1790 IF C3=2 THEN GOSUB 2930
1800 IF C3=3 THEN PO=3:GOSUB 3040
1810 NEXT I
1820 TYPE=0
1830 GOSUB 2070
1840 PRINT #NP,"sp0;"
1850 CLOSE: END
1860 'Open # NP as plot output
1870 '=====================
1880 OPEN "com1:9600,s,7,1,RS,CS65535,DS,CD" AS # NP
1890 PRINT # NP,"IN;"
1900 RETURN
1910 'Get pen # PN
1920 '=====================
1930 PRINT # NP, "SP ";PN;";"
1940 RETURN
1950 'Set pen velocity to PENSPEED
1960 '=====================
1970 PRINT # NP,"VS";PENSPEED;";"
1980 RETURN
1990 'Move pen to X,Y
2000 '=====================
2010 PRINT # NP,"PA";X;Y;";"
2020 RETURN
2030 'Pen down
2040 '=====================
2050 PRINT # NP,"PD;"
2060 RETURN
2070 'Pen UP
2080 '=====================
2090 PRINT # NP,"PU;"
2100 RETURN
2110 'Write TIT$ at X,Y of set CL
2120 '=====================
2130 GOSUB 2070
2140 GOSUB 1990
2150 PRINT #NP,"CP";CL;CD;";"
```

```
2160 CD=0:CL=0
2170 PRINT # NP,"LB";TIT$;CHR$(3)
2180 RETURN
2190 'Set character height (CH) and width (CW)
2200 '=====================
2210 PRINT # NP,"SI";CW;CH;";"
2220 RETURN
2230 'Rotate coordinate system RO degrees (0 or 90)
2240 '=====================
2250 PRINT # NP,"RO";RO;";"
2260 RETURN
2270 'Set character run (CX) and rise (CY)
2280 '=====================
2290 PRINT # NP,"DI";CX;CY;";"
2300 RETURN
2310 'Plot circle at X,Y with radius RC and cord angle CC
2320 '=====================
2330 GOSUB 1990
2340 PRINT # NP,"CI";RC;CC;";"
2350 RETURN
2360 'Plot box at X,Y with height BH and width BW
2370 '=====================
2380 GOSUB 2070
2390 GOSUB 1990
2400 GOSUB 2030
2410 PRINT # NP,"EA";X+BW;Y+BH;";"
2420 GOSUB 2070
2430 RETURN
2440 'Fill box at X,Y with height BH and width BW
2450 '=====================
2460 GOSUB 1990
2470 PRINT # NP,"RA";X+BW;Y+BH;";"
2480 RETURN
2490 'Fill TYPE =1 & 2 (Solid), 3 (Lines), 4,5 & 6 (Cross-hatch)
2500 '=====================
2510 DD=0:BA=0:IF TYPE=3 THEN DD=100
2520 IF TYPE=4 THEN DD= 50: BA=45
2530 IF TYPE=5 THEN DD= 50: BA=90
2540 IF TYPE=6 THEN DD= 50: BA=135
2550 PRINT # NP,"FT";TYPE;DD;BA;";"
2560 RETURN
2570 'Select new pen
2580 '=====================
2590 PN=(PN MOD 6) +1
2600 GOSUB 1910
```

```
2610 RETURN
2620 'Draw box at X,Y to X1,Y1
2630 '=====================
2640 GOSUB 1990:GOSUB 2040
2650 PRINT # NP,"PA";X1;Y;X1;Y1;X;Y1;X;Y;";"
2660 GOSUB 2070
2670 RETURN
2680 'radium variation
2690 F$=TIME$:G$=RIGHT$(F$,2)+LEFT$(F$,2)
2700 T=VAL(G$) MOD 501
2710 LOCATE 10,50:PRINT F$;T
2720 FOR K=1 TO T:L=RND
2730 NEXT K:SA=INT(RND*3.5+12):SB=INT(RND*7.9):SC=(SB+RND*7) MOD 7
2740 COLOR SA,SB,SC
2750 RETURN
2760 'Write NUMB at X,Y of set CL
2770 '=====================
2780 HR=0
2790 TIT$=STR$(NUMB)
2800 LN=LEN(TIT$)
2810 IF LN<7 THEN 2900
2820 IF HR=1 THEN 2840
2830 NUMB=INT(NUMB*1000+.499)/1000:HR=1:GOTO 2790
2840 '
2850 AA=NUMB-INT(NUMB): BB=1-AA
2860 IF AA<1 THEN 2900
2870 IF 10*AA >1 THEN 2890
2880 BB=BB/10
2890 ZZ=INT(NUMB*1000+BB*1000)/1000:NUMB=ZZ:TIT$=STR$(ZZ)
2900 IF LN<6 THEN TIT$=STRING$(6-LN," ")+TIT$
2910 GOSUB 2110
2920 RETURN
2930 'Do a bar form 0 to Y that is DEL-X wide
2940 '=====================
2950 IF I<>MAX THEN DEL = BX*(X(I+1)-X(I))
2960 IF I=MAX THEN DEL = BX*(X(I)-X(I-1))
2970 BH=AY-Y: BW=-DEL
2980 X=X+DEL*.5
2990 GOSUB 2360
3000 IF TYPE<2 THEN 3030
3010 GOSUB 2490
3020 GOSUB 2440
3030 RETURN
3040 'Plot point of type PO
3050 '=====================
```

```
 3060 PSZ=TICY:IF TICX<PSZ THEN PSZ=TICX
 3070 IF PO<3 THEN 3140
 3080 X=X+2*TICX: GOSUB 1990
 3090 GOSUB 2030
 3100 X=X-3*TICX:Y=Y-TICY:GOSUB 1990
 3110 Y=Y+2*TICY:GOSUB 1990
 3120 X=X+3*TICX:Y=Y-TICY:GOSUB 1990
 3130 GOTO 3150
 3140 GOSUB 1990: CC=120-30*PO: RC=PSZ: GOSUB 2030:GOSUB 2310
 3150 GOSUB 2070: RETURN
50000 'Display and select item from menu
50010 '
50020 START=4:YR=2:XR=23:SIZ=14:PAG=1:Q=1
50030 DAG$="Return to select"+CHR$( 32)+"--"+CHR$(26)
50040 DIG$=DIR$(START):DOG$="Input file name from key board "
50050 DIR$(START)=DOG$
50060 T=VAL(RIGHT$(TIME$,2)):FOR I=1 TO T:C1=RND*7.99:NEXT I
50070 C2=INT(C1):C1=C2+8:C3=C1:IF C2=0 THEN C1=7:IF RND>.5 THEN
C1=15:C2=0
50080 COLOR C1,C2,C3:CLS:C1T=15:C2T=0:IF C1=C1T THEN C1T=0:C2T=7
50090 LOCATE YR-1,XR:COLOR C1T,C2T:PRINT " Files to select from ";
50100 LOCATE 2, 2:PRINT " To move up use   ";CHR$(25);" ";
50110 LOCATE 3, 2:PRINT " or PgUp          ";
50120 LOCATE 4, 2:PRINT " To move down use ";CHR$(24);" ";
50130 LOCATE 5, 2:PRINT " or PgDn          ";
50140 COLOR C1,C2
50150 FOR I=2 TO START-1
50160 LOCATE YR+I-2,XR:PRINT DIR$(I);
50170 NEXT I
50180 LS=START:LX=START+SIZ:TAG=LS:LM=LS
50190 CAN=SIZ+PAG+START
50200 LOCTS=YR+Q+TAG MOD CAN
50210 IF LS<SIZ+START THEN Q=1
50220 IF LS<LM THEN LS=LM
50230 FOR I=LS TO LX
50240 YL=YR+Q+I MOD CAN
50250 LOCATE YL,1
50260 PRINT STRING$(XR-1," ");
50270 PRINT DIR$(I);
50280 LD=LEN(DIR$(I)):IF (LD+XR)<79 THEN PRINT STRING$(79-LD-XR," ");
50290 NEXT I
50300 COLOR C1,C2
50310 LOCT=YR+Q+ TAG MOD CAN:IF E>0 THEN LOCTS=LOCT
50320 LOCATE LOCTS,1:PRINT STRING$(79," ");
50330 E=0
```

```
50340 LOCATE LOCTS,XR:PRINT DIR$(TAGS);
50350 COLOR C1T+16,C2T:TAGS=TAG:LOCTS=LOCT
50360 LOCATE LOCT,2:PRINT DAG$;:COLOR C1T,C2T
50370 LOCATE LOCT,XR:PRINT DIR$(TAG);
50380 T$=INKEY$:IF T$="" THEN 50380
50390 A1=ASC(T$):TL=LEN(T$):A2=ASC(RIGHT$(T$,1))
50400 IF TL>1 AND A2=80 THEN TAG=TAG+1
50410 IF TL>1 AND A2=72 THEN TAG=TAG-1
50420 IF TL>1 AND A2=81 AND TAG=LX THEN TAG=LX + 1:GOTO 50450
50430 IF TL>1 AND A2=81 THEN TAG=LX
50440 IF TL>1 AND TAG=LS AND A2=73 THEN TAG=LS   -1:GOTO 50460
50450 IF TL>1 AND A2=73 THEN TAG=LS
50460 IF A1=13 THEN 50510
50470 IF TAG<LM THEN TAG=LM
50480 IF TAG>LX THEN LS=TAG :LX=LS+CAN-1:E=TAG:Q=START+1:COLOR C1,C2:GOTO
50210
50490 IF TAG<LS THEN LX=TAG:LS=LX-CAN+1:E=TAG :COLOR C1,C2:GOTO 50210
50500 GOTO 50300
50510 DIR$(START)=DIG$
50520 DEFILE$=LEFT$(DIR$(TAG),12)
50530 IF LEN(DEFILE$)<9 THEN 50580
50540 MID$(DEFILE$,9)="."
50550 IF TAG<=START OR TAG>IX THEN DEFILE$=""
50560 IF DEFILE$<>"" THEN GOSUB 50630:DEFILE$=FRONTPT$+DEFILE$
50570 RETURN
50580 CLS:INPUT" Name of file = ";DEFILE$:IF DEFILE$<>"" THEN RETURN
50590 FOR I=1 TO 2
50600 BEEP:BEEP:BEEP:PRINT " You need select or input a file to work on ! " 50610
NEXT I:GOTO 50000
50620 '
50630 ' Find drive and subdir.
50640 FRONTPT$=DIR$(3):IFP=INSTR(FRONTPT$,":")
50650 IF IFP<1 THEN FRONTPT$="":RETURN
50660 LFQ=LEN(FRONTPT$)
50670 FRONTPT$=RIGHT$(FRONTPT$,LFQ-IFP+2)
50680 IFQ=INSTR(FRONTPT$,"\"):LFR=LEN(FRONTPT$)
50690 IF IFQ=LFR THEN FRONTPT$=LEFT$(FRONTPT$,LFR-1)
50700 '
50710 RETURN
50720 CLOSE
50730 OPEN "dedir.tmp" FOR INPUT AS 1
50740 FOR I=1 TO 512:IF EOF(1) THEN 50760
50750 LINE INPUT #1,DIR$(I):NEXT I
50760 IX=I-1
50770 CLOSE
50780 RETURN
^Z
```

[BLANK PAGE]

APPENDIX J.37

MONTE.BAS

```
 10 width "lpt1:",132
 20 rr$=right$(time$,2):for i=1 to val(rr$):z=z+z/(z+1)+RND:next i
 30 ff=.1468:A00=187:LPRINT "Per ";a00; " Aircraft "
 40 input "Do you want plot output (Y/N) ";d$
 50 d$=left$(d$,1)
 60 if d$<>"Y" and d$<>"y" then 150
 69 iwp=1
 70 input" Name of plot file       = ";fpl$
 80 input" Title at top    of plot = ";t1$
 90 input" Title at side   of plot = ";t2$
100 input" Title at bottom of plot = ";t3$
110 open fpl$ for output as 3
120 print #3,t1$
130 print #3,t2$
140 print #3,t3$
150 input "NUMBER OF DAYS   ";DAYS
160 input "MTBF             ";MTBF
170 input "MODE             ";MODE
180 input "BUY PERIOD(DAYS) ";PER
190 input "FACTOR           ";F
200 dim t(200),c(200),s(200),g(1500),W(200)
210 input "Name of file = ";fi$
220 open fi$ for input as 1
230 for i=1 to 3
240 input #1,a$
250 next i
260 max=1500
270 for i=1 to max
280 if eof(1) then 310
290 input #1,t(i),c(i),s(i),d
300 next i
310 max=i-1
320 '
330 for i=1 to DAYS
340 IF (I MOD PER)=1 THEN IV=(1/MTBF + F/MODE/I)*A00*PER +IV
350 rt=rnd
360 for j=1 to max:if rt-s(j) <0 then 380
370 next j
380 g=t(j)+1
390 BOT=S(J)-S(J-1):IF BOT<=0 THEN BOT=1
```

```
400 G=G + (T(J)-T(J-1))*(RT-S(J))/BOT
410 W(J)=W(J)+1: TJ=TJ+1
420 nf=A00/g
430 nf=nf*ff
440 iv=iv-nf
450 if (i mod 10)= 1 then lprint
460 ctf=ctf+nf:anf=ctf/i
470 lprint using "#### ##.# ";ctf,anf;
480 g(i)=iv
490 next i
500 lprint
510 FOR J=1 TO MAX: SC=W(J)/TJ+SC
520 bb=s(j)-sc
530 LPRINT USING "#### #.## #.## #.## "; T(J),S(J),SC,bb
540 NEXT J
550 for i=1 to days
560 if (i mod 10)= 1 then lprint
570 '
580 lprint using "#### #### ";(i),g(i);
590 if g(i)<0 then daysu=daysu+1
591 if iwp=1 then print #3,i,g(i),zzz,zzz
600 next i
610 lprint:lprint:lprint "Number of days that Inv. was depleted = ";daysu
620 lprint "Percent of the time Inv. was depleted = ";daysu/days*100
630 close:end
^Z
```

APPENDIX J.38

STATDIS5.BAS

```
  2 DIM DIR$(512):MAX=512:color 0,7:cls:gosub 50720
 10 CLOSE
 20 SNO$="1234567890":STN$="          1          2          3          4          5
6          7          8          9"
 30 SNO$=SNO$+SNO$:SNN$=SNO$+SNO$+SNO$+SNO$
 40 WIDTH "lpt1:",132
 50 X0=10
 60 DIM D(300,5),S(300,5),A(5,5)
 70 LPRINT " f(t) = A1/t*exp[A3*(ln(t) + A2)^2] "
 80 INPUT "Top of plot    = ";T1$
 90 INPUT "Side of plot   = ";T2$
100 INPUT "Bottom of plot = ";T3$
110 PRINT "-------------------------------------- "
120 PRINT "                    "
130 INPUT "FILE NAME =";FI$
133 if fi$<>"" then 140
135 gosub 50000:fi$=defile$
140 XX=LEN(FI$):YY=INSTR(FI$,"."):IF YY=0 THEN 160
150 XX=YY
160 FP$=LEFT$(FI$,XX)+"plt"
170 OPEN FI$ FOR INPUT AS 1
180 OPEN FP$ FOR OUTPUT AS 3
190 PRINT #3,T1$
200 PRINT #3,T2$
210 PRINT #3,T3$
220 INPUT #1,V
230 V1=V-1
240 IF EOF(1) THEN 370
250 INPUT #1,F
260 IF EOF(1) THEN 370
270 INPUT #1,D
280 ID=INT(D/365)+1
290 FOR I=1 TO V1
300 IF EOF(1) THEN 370
310 INPUT #1,X
320 NEXT I
330 I=INT(F/X0+.5)+1
340 D(I,ID)=D(I,ID)+1:T=T+1
350 D(I,0)=D(I,0)+1
360 GOTO 240
```

```
370 'CONTINUE
380 FOR I=1 TO 150
390 IF D(I,0)=0 THEN ZN=ZN+1!
400 NEXT I
410 ZF=.499/ZN
420 FOR J=0 TO 4
430 F1=0!:F2=0!:F3=0!:F4=0!:F5=0!:S0=0:S1=0!:S2=0!:S3=0!:S4=0!:S5=0!:S6=0!:S7=0!
440 '
450 FOR I=1 TO 150:TT=I*X0
460 IF D(I,J)=0 THEN 510
470 T1=LOG(TT):T2=LOG(D(I,J)*TT):T3=LOG(D(I,J)/TT^K):T4=T1*T1
480 F1=F1+T2: F2=F2+T1*T2: F3=F3+T2*T4: F4=F4+T1*T3 : F5=F5+T3
490 S0=S0+1: S1=S1+T1: S2=S2+T4: S3=S3+T4   : S4=S4+T4*T1: S5=S5+T4*T4
500 '
510 NEXT I
520 R1=S1*F1-S0*F2:R2=S1*S1-S0*S3:R3=S1*S2-S4*S0
530 R4=S2*F2-S1*F3:R5=S2*S3-S1*S4:R6=S2*S4-S1*S5
540 A3=0:BOT=R3*R5-R6*R2:IF BOT<>0 THEN A3=(R1*R5-R4*R2)/BOT
550 A2=0:IF R2<>0 THEN A2=(R1-A3*R3)/R2
560 A1=(F1-A2*S1-A3*S2)/S0
570 A(3,J)=A3:A(2,J)=A2*.5/A3:A(1,J)=EXP(A1-A3*A(2,J)^2):A(4,J)=(A4):A(5,J)=A5
580 LPRINT "j = ";J;"a1 = ";A(1,J);" a2 = ";A(2,J);" a3 = ";A(3,J)
590 '
600 NEXT J
610 FOR J=0 TO 4:LPRINT CHR$(12);" Year = ";J:LPRINT" O=log normal fit "
620 LPRINT "                    |---- Number of failures in period ----------->"
630 LPRINT "     Obs.     Cal.    ";STN$
640 LPRINT " Tbf  No.      No.     ";SNO$+SNN$
650 TD=0:TF1=0:TF2=0
660 FOR I=1 TO 150
670 S(I,J )=D(I,J )/T
680 NEXT I
690 CC=0
700 A1=A(1,J):A2=A(2,J):A3=A(3,J)
710 FOR I=1 TO 150
720 X=I*X0 : LX= LOG(X)
730 C=A1/X*EXP(A3*(LX+A2)^2): CC=CC+C/T
740 S(I,J)=S(I,J)+S(I-1,J)
750 LPRINT USING "#### ### #.## ### #.## ";X;D(I,J);S(I,J);C;CC;
760 if j=0 then PRINT #3,X;D(I,J);S(I,J);C
770 TD=TD+D(I,J):TF1=TF1+C
780 X8=D(I,J):X9=100-X8:IF X9<1 THEN X9=1
790 DZ=D(I,J):IF DZ>100 THEN DZ=100
800 CZ=INT(C)+1:IF CZ>100 THEN CZ=100
810 SW$=   STRING$(DZ    ,"*")+STRING$(X9,"-"):MID$(SW$,CZ       ,1)="O"
820 LPRINT SW$
```

```
  830 NEXT I
  840 LPRINT        "    ====     ====   "
  850 LPRINT USING "    ####     ####   ";TD;TF1
  860 NEXT J
50000 'Display and select item from menu
50010 '
50020 START=4:YR=2:XR=23:SIZ=14:PAG=1:Q=1
50030 DAG$="Return to select"+CHR$( 32)+"--"+CHR$(26)
50040 DIG$=DIR$(START):DOG$="Input file name from key board "
50050 DIR$(START)=DOG$
50060 T=VAL(RIGHT$(TIME$,2)):FOR I=1 TO T:C1=RND*7.99:NEXT I
50070 C2=INT(C1):C1=C2+8:C3=C1:IF C2=0 THEN C1=7:IF RND>.5 THEN C1=15:C2=0
50080 COLOR C1,C2,C3:CLS:C1T=15:C2T=0:IF C1=C1T THEN C1T=0:C2T=7
50090 LOCATE YR-1,XR:COLOR C1T,C2T:PRINT " Files to select from ";
50100 LOCATE 2, 2:PRINT " To move up use   ";CHR$(24);" ";
50110 LOCATE 3, 2:PRINT " or PgUp          ";
50120 LOCATE 4, 2:PRINT " To move down use ";CHR$(25);" ";
50130 LOCATE 5, 2:PRINT " or PgDn          ";
50140 COLOR C1,C2
50150 FOR I=2 TO START-1
50160 LOCATE YR+I-2,XR:PRINT DIR$(I);
50170 NEXT I
50180 LS=START:LX=START+SIZ:TAG=LS:LM=LS
50190 CAN=SIZ+PAG+START
50200 LOCTS=YR+Q+TAG MOD CAN
50210 IF LS<SIZ+START THEN Q=1
50220 IF LS<LM THEN LS=LM
50230 FOR I=LS TO LX
50240 YL=YR+Q+I MOD CAN
50250 LOCATE YL,1
50260 PRINT STRING$(XR-1," ");
50270 PRINT DIR$(I);
50280 LD=LEN(DIR$(I)):IF (LD+XR)<79 THEN PRINT STRING$(79-LD-XR," ");
50290 NEXT I
50300 COLOR C1,C2
50310 LOCT=YR+Q+ TAG MOD CAN:IF E>0 THEN LOCTS=LOCT
50320 LOCATE LOCTS,1:PRINT STRING$(79," ");
50330 E=0
50340 LOCATE LOCTS,XR:PRINT DIR$(TAGS);
50350 COLOR C1T+16,C2T:TAGS=TAG:LOCTS=LOCT
50360 LOCATE LOCT,2:PRINT DAG$;:COLOR C1T,C2T
50370 LOCATE LOCT,XR:PRINT DIR$(TAG);
50380 T$=INKEY$:IF T$="" THEN 50380
50390 A1=ASC(T$):TL=LEN(T$):A2=ASC(RIGHT$(T$,1))
50400 IF TL>1 AND A2=80 THEN TAG=TAG+1
```

```
50410 IF TL>1 AND A2=72 THEN TAG=TAG-1
50420 IF TL>1 AND A2=81 AND TAG=LX THEN TAG=LX + 1:GOTO 50450
50430 IF TL>1 AND A2=81 THEN TAG=LX
50440 IF TL>1 AND TAG=LS AND A2=73 THEN TAG=LS   -1:GOTO 50460
50450 IF TL>1 AND A2=73 THEN TAG=LS
50460 IF A1=13 THEN 50510
50470 IF TAG<LM THEN TAG=LM
50480 IF TAG>LX THEN LS=TAG :LX=LS+CAN-1:E=TAG:Q=START+1:COLOR C1,C2:GOTO
50210
50490 IF TAG<LS THEN LX=TAG:LS=LX-CAN+1:E=TAG :COLOR C1,C2:GOTO 50210
50500 GOTO 50300
50510 DIR$(START)=DIG$
50520 DEFILE$=LEFT$(DIR$(TAG),12)
50530 IF LEN(DEFILE$)<9 THEN 50580
50540 MID$(DEFILE$,9)="."
50550 IF TAG<=START OR TAG>IX THEN DEFILE$=""
50560 IF DEFILE$<>"" THEN GOSUB 50630:DEFILE$=FRONTPT$+DEFILE$
50570 RETURN
50580 CLS:INPUT" Name of file = ";DEFILE$:IF DEFILE$<>"" THEN RETURN
50590 FOR I=1 TO 2
50600 BEEP:BEEP:BEEP:PRINT " You need select or input a file to work on ! "
50610 NEXT I:GOTO 50000
50620 '
50630 ' Find drive and subdir.
50640 FRONTPT$=DIR$(3):IFP=INSTR(FRONTPT$,":")
50650 IF IFP<1 THEN FRONTPT$="":RETURN
50660 LFQ=LEN(FRONTPT$)
50670 FRONTPT$=RIGHT$(FRONTPT$,LFQ-IFP+2)
50680 IFQ=INSTR(FRONTPT$,"\"):LFR=LEN(FRONTPT$)
50690 IF IFQ=LFR THEN FRONTPT$=LEFT$(FRONTPT$,LFR-1)
50699 if ifq<lfr then frontpt$=frontpt$+"\"
50700 '
50710 RETURN
50720 CLOSE
50730 OPEN "dedir.tmp" FOR INPUT AS 1
50740 FOR I=1 TO 512:IF EOF(1) THEN 50760
50750 LINE INPUT #1,DIR$(I):NEXT I
50760 IX=I-1
50770 CLOSE
50780 return
^Z
```

DISTRIBUTION LIST

Internal Distribution

1. E. L. Allred
2. E. D. Arnold
3-5. H. W. Bertini
6. R. W. Bonnett
7. R. J. Borkowski
8. J. R. Buchanan
9. W. D. Burch
10. D. W. Burton
11. S. Cantor
12. R. A. Cooper, Jr.
13. W. L. Cooper
14. W. K. Crowley
15. G. A. Dailey
16-20. R. C. DeLozier
21. D. M. Eissenberg
22. B. R. Everman
23. E. C. Fox
24. C. P. Frew
25. W. Fulkerson
26. J. A. Getsi
27. R. W. Glass
28. R. O. Green
29. P. M. Haubenreich
30. R. F. Holdaway
31-35. D. A. Holder
36. V. C. Huffstetler
37. R. A. Kisner
38. H. E. Knee
39. R. P. Leinius
40. J. W. McKeever
41. F. C. Maienschien
42. J. T. Manning
43. G. T. Mays
44. R. J. Moses
45. G. A Murphy
46. S. C. Nelson, Jr.
47. L. C. Oakes
48. R. D. Oglesby
49. B. R. Rodgers
50. R. M. Schillling, Jr.
51. J. L. Sexton
52. R. H. Thornton
53. R. O. Thornton
54. H. E. Trammell
55. V. K. Wilkinson
56. ORNL Patent Office
58. C & TD Library
58-59. Central Research Library
60. Document Reference Section
61. Laboratory Records Department
62. Laboratory Records (RC)

External Distribution

63-77. Fred Fernandez, Naval Aviation Logistics Center (Code 631), Patuxent River, MD 20670-5449
78-80. Jeff Keates, Naval Aviation Logistics Center (Code 631), Patuxent River, MD 20670-5449
81. Jennifer Lee, Naval Aviation Logistics Center (Code 631), Patuxent River, MD 20670-5449
82-84. L. E. Rowland, Naval Aviation Logistics Center (Code 631), Patuxent River, MD 20670-5449
85. Office of Assistant Manager for Energy Research and Development, Department of Energy, ORO, Oak Ridge, TN 37831
86-115. Technical Information Center, Department of Energy, Oak Ridge, TN 37831