Defense Nuclear Agency
Alexandria, VA 22310-3398

DNA-TR-92-105

Electromagnetic Systems Effects Database (EMSED)

AERO 90 Phase II User's Manual

Kalim A. Sawires
TRW S. I. G.
Systems Engineering & Development Division
P.O. Box 6213
Carson, CA 90746

February 1998

Technical Report

CONTRACT No. DNA 001-87-C-0274

DESTRUCTION NOTICE:

Destroy this report when it is no longer needed.
Do not return to sender.

PLEASE NOTIFY THE DEFENSE SPECIAL WEAPONS AGENCY, ATTN: CSTI, 6801TELEGRAPH ROAD, ALEXANDRIA, VA 22310-3398, IF YOUR ADDRESS IS INCORRECT, IF YOU WISH IT DELETED FROM THE DISTRIBUTION LIST, OR IF THE ADDRESSEE IS NO LONGER EMPLOYED BY YOUR ORGANIZATION.

CUT HERE AND RETURN

DISTRIBUTION LIST UPDATE

This mailer is provided to enable DSWA to maintain current distribution lists for reports. (We would appreciate your providing the requested information.)

- ☐ Add the individual listed to your distribution list.
- ☐ Delete the cited organization/individual.
- ☐ Change of address.

NOTE:
Please return the mailing label from the document so that any additions, changes, corrections or deletions can be made easily. For distribution cancellation or more information call DSWA/IMAS (703) 325-1036.

NAME: ____________________

ORGANIZATION: ____________________

OLD ADDRESS	**CURRENT ADDRESS**
____________________	____________________
____________________	____________________
____________________	____________________

TELEPHONE NUMBER: (   ) ____________________

DSWA PUBLICATION NUMBER/TITLE	**CHANGES/DELETIONS/ADDITIONS, etc.)** *(Attach Sheet if more Space is Required)*
____________________	____________________
____________________	____________________
____________________	____________________

DSWA OR OTHER GOVERNMENT CONTRACT NUMBER: ____________________

CERTIFICATION OF NEED-TO-KNOW BY GOVERNMENT SPONSOR (if other than DSWA):

SPONSORING ORGANIZATION: ____________________

CONTRACTING OFFICER OR REPRESENTATIVE: ____________________

SIGNATURE: ____________________

DEFENSE SPECIAL WEAPONS AGENCY
ATTN: IMAS
6801 TELEGRAPH ROAD
ALEXANDRIA, VA 22310-3398

DEFENSE SPECIAL WEAPONS AGENCY
ATTN: IMAS
6801 TELEGRAPH ROAD
ALEXANDRIA, VA 22310-3398

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	980201	Technical 871231 - 970101

4. TITLE AND SUBTITLE
Electromagnetic Systems Effedts Database (EMSED) Aero 90, Phase II - User's Manual

5. FUNDING NUMBERS
C - DNA 001-87-C-0274
PE - 62715H
PR - RG
TA - RE
WU - DH00008

6. AUTHOR(S)
Kalim A. Sawires

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
TRW S. I. G.
Systems Engineering & Development Division
P. O. Box 6213
Carson, CA 90746

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Special Weapons Agency
6801 Telegraph Road
Alexandria, VA 22310-3398
ESE/Rooney

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
DNA-TR-92-105

11. SUPPLEMENTARY NOTES
This work was sponsored by the Defense Special Weapons Agency under RDT&E RMC Code B 4662 D RG RE 00008 RAEE 3260 A 25904D.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The Electromagnetic Systems Effects Database (EMSED), also called AIRBASE, is a training guide for users not familiar with the AIRBASE database and its operating platform, the Macintosh computer (Mac). The objectives are to efficiently archive EMP test data and provide a useful signal processing and analysis tool for future EMP studies.

The first chapter is an introduction to the hardware and software used for this program. This manual will guide the novice through the basics of the Mac operating system in Chapter 2. The use will also acquire knowledge in the basics of the AIRBASE database, and learn how to search, retrieve, and analyze data. A basic tutorial of 4th DIMENSION is included in Chapter 3, and MATLAB starter is provided in Chaptaer 4. The complete structure and contents of AIRBASE are discussed in Chapter 5.

The AIRBASE database was developed under a commercial product called 4th DIMENSION to minimize development and update costs, while retaining the power of a fully relational database architecture.

14. SUBJECT TERMS
EMSED Airbase
Database Electromagnetic Pulse

15. NUMBER OF PAGES
106

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Sta. 239-18
298-102

CLASSIFIED BY:	N/A since Unclassfied.
DECLASSIFY ON:	N/A since Unclassified.

Contents

Chapter 1

Introduction

Welcome to the Electromagnetic System Effects Database (EMSED), also called AIRBASE[1]. Throughout the rest of this manual we will refer to the system as AIRBASE, to avoid confusion. This manual is intended as a training guide for users *not* familiar with the AIRBASE database and its operating platform, the Macintosh computer (Mac). The objectives are to efficiently archive EMP test data and provide a useful signal processing and analysis tool for future EMP studies.

The first chapter is an introduction to the hardware and software used for this program. This manual will guide the novice through the basics of the Mac operating system in Chapter 2. The user will also acquire knowledge in the basics of the AIRBASE database, and learn how to search, retrieve, and analyze data. A basic tutorial of 4th Dimension is included in Chapter 3, and MATLAB starter is provided in Chapter 4. The complete structure and contents of AIRBASE are discussed in Chapter 5.

The AIRBASE database was developed under a commercial product called 4th DIMENSION to minimize development and update costs, while retaining the power of a fully relational database architecture.

1.1 AERO 90 Phase II Objectives

The AERO 90 Phase II program objective was to give the Defense Special Weapons Agency (DSWA) tools to perform data analysis of the objects in the AIRBASE database, as well

[1]AIRBASE is the original product developed for Field Command Defense Nuclear Agency (FCDNA) to support tests taken in the 1980s at Air Force Weapons Laboratory

as add new test objects into the database in a consistent fashion. Two copies of the High Altitude EMP system effects workstation will be delivered to DSWA upon completion of this project[2].

The AERO 90 Phase II program consists of three primary tasks. These tasks are as follows:

1. TRW will accept the High Altitude EMP Effects Database[3] from Lawrence Livermore National Laboratory, and load this data into a Macintosh workstation in a format compatible with the AIRBASE-style.

2. TRW will help train designated DNA project officers and contractors in the use of the database.

3. Any software developed to implement database applications shall be documented in a user handbook.

1.2 Computer Configuration

The AERO 90 computer is a Macintosh workstation equipped with a relational database, a spreadsheet, signal processing software, a graphics package, a C programming package, and a viewgraph display generator.

1.2.1 Hardware

The Macintosh hardware is as follows:

- Macintosh IIfx
  - 40 MHz 68030 CPU with 68882 math coprocessor
  - 6 nubus expansion slots
  - SCSI (Small Computer System Interface) port
  - Printer port

[2]The exact locations of these workstations is still to be determined, but as of the date of this publication, one of the workstations is located at the DASIAC (Department of Defense Nuclear Information Analysis Center) facility in Alexandria, VA, and the other is scheduled to reside at headquarters DSWA

[3]Developed in a joint effort between LLNL and DSWA

  - Modem port

- 13" Apple color monitor
- 8 bit Apple color graphics card
- 650 Megabyte Rodime external hard drive
- 210 Megabyte Rodime external hard drive
- 44 Megabyte Mass Microsystems removable Syquest-type disk drive
- Macintosh enhanced keyboard II
- Macintosh 1-button mouse

1.2.2 Software

The AERO 90 software list is as follows:

- Macintosh System 7 Software – operating system for the Macintosh
- 4th DIMENSION 2.1 – relational database management system
- Microsoft Word 4.0 – word processing package
- Microsoft PowerPoint 2.1 – viewgraph creator
- MATLAB 3.5 – digital signal processing and data analysis package
- Canvas 3.0 – multi-purpose graphics package
- WingZ 1.1 – spreadsheet and plotting package
- Think C – programming language package

Chapter 2

Macintosh Fundamentals

2.1 MACINTOSH BASICS

The Macintosh computer was selected because it has an intuitive graphical user interface. This chapter is intended as a quick-start guide to operating the Macintosh, not a comprehensive tutorial. The details and finer points of the operating system can be found in the Macintosh System reference manual [2].

2.1.1 Turning the Computer On and Off

The ON button of a Macintosh IIfx is located on keyboard near the upper right corner. Turn on the computer now. In a few seconds the screen display termed the *desktop*, similar to that shown in Figure 2.1 will appear. To turn *off* this computer select the **Shutdown** command from the **Special** menu. Menus and commands will be explained in the following sections.

2.1.2 Mouse/Pointer Relationship

All Macintosh computers are equipped with a one-button mouse. The mouse is an inseparable part of computer operation. To use a mouse, place it on a flat surface and move it around while viewing the monitor. Notice a *pointer*(see Figure 2.2) that moves on the monitor in conjunction with the movement of the mouse on a flat surface .

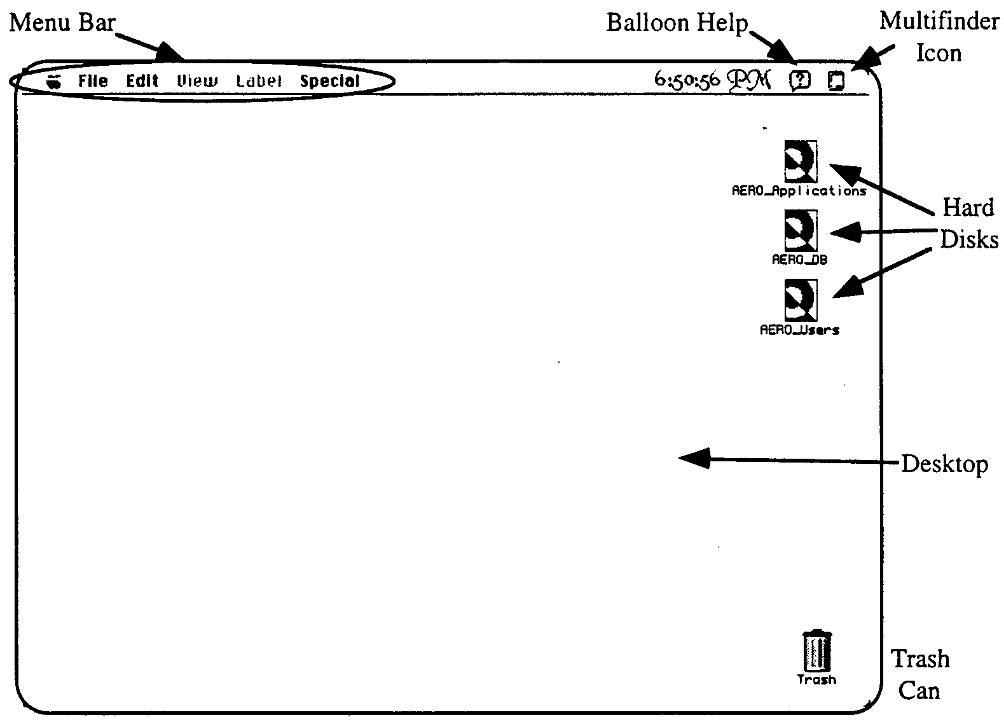


Figure 2.1: Macintosh Desktop - labelled

Figure 2.2: Macintosh Pointer

2.1.3 Mouse Clicking

Position the pointer on the Aero_Applications icon (a small picture on the top right portion of the desktop), then depress and quickly release the mouse button. This term is referred to as a mouse *click* or a *single-click*. A *double-click* is two single-clicks performed rapidly in succession.

To move icons another action with the mouse is used. Position the mouse on the Aero_Applications icon and press and hold the mouse button down. With the mouse button still depressed, *drag* the pointer to another part of the desktop. The Aero_Applications icon was moved and the action is referred to as *dragging*. Any icon can be moved in this fashion (try moving the Trash Can icon to a different part of the desktop). Now place the icons back in their original positions. For more detailed information on the use of the mouse, refer to reference [2, pp 2–6].

2.1.4 Special Keys

Throughout this text we will be referring to three special keys for certain operations. They are all located horizontally in line with the "space" bar. The one labelled **control** is obviously the control key. Two such keys exist, both on the far left and far right of the standard keyboard. The next key is the **option** key. Finally, the **command** key is located on either side of the "space" bar. This key is marked by an apple with a bite taken out, as well as a "four-leaf clover" shape.

2.1.5 Choosing Menu Commands

The menu bar is located at the top of the Macintosh monitor (as shown in Figure 2.1). All applications contain a menu bar, but the functions contained in each may vary greatly. Three of the menus are consistent between all applications. These are the **Apple** menu, **File** menu, and **Edit** menu. The **Apple** menu (leftmost on the menu bar) stores desk accessories (explained later in this chapter). The **File** menu performs basic file operations (opening, closing, saving, quitting, etc.) The **Edit** menu contains the very useful *Cut, Copy, Paste* and *Undo* commands.

A command can be chosen from a menu by performing actions similar to the following sequence of steps:

1. Select the Aero_Applications icon by moving the pointer to the hard disk and single-

clicking. The colors in the icon will now be inverted in appearance, or highlighted. If the window is already open, close it by clicking the "close box" in the top left title bar for the window shown in Figure 2.3.

2. Move the mouse to the **File** menu, press and *hold the mouse down.* A list of commands that are available are shown under the menu. The ones that are active are solid black, while the inactive commands are grayed-out.

3. With the mouse still depressed, choose **Open** from the list of commands by *dragging* the pointer down the list and releasing the button when the **Open** command appears as inverse video. The Aero_Applications Icon will open, displaying the contents of the hard disk (See Figure 2.3).

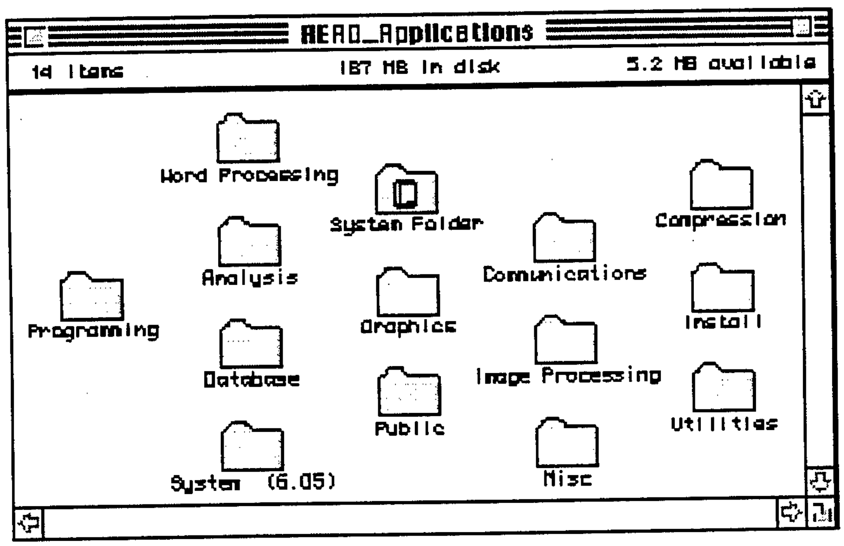


Figure 2.3: Contents of Aero_Applications

2.1.6 Manipulating a Window

This topic refers to Figure 2.4. The more important features of a Macintosh window are discussed in this section.

1. Move the mouse to the *grow* box of the open window; then press the mouse down and drag the grow box as you did with the icon earlier. This action varies the size of the window.

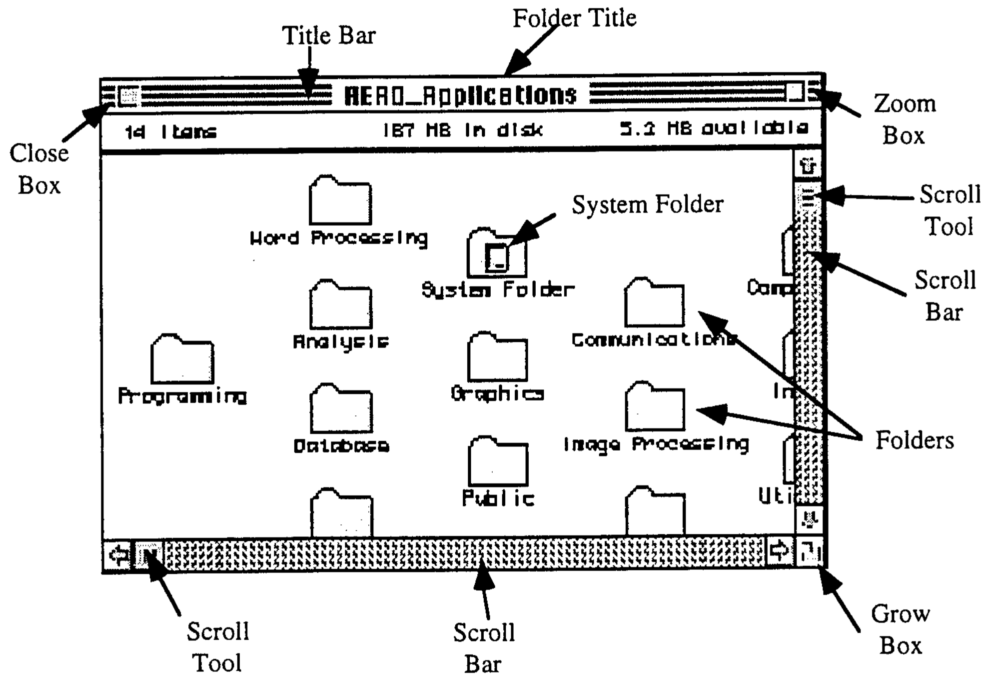


Figure 2.4: The Macintosh Window - labelled

2. Now single-click on the *zoom* box. This modifies the size of the open window to display all of the icons visible in that folder, if possible.

3. Click on the zoom box once more. Note that the window reverts to its original size and placement.

4. Now close the window by single-clicking on the *close* box. The window has been "put away".

5. To reopen the window double-click on the Aero_Applications icon

Take a moment to look over the menu bar, located at the top of the monitor. **Apple** menu items can be accessed from the **Apple** menu. Desk accessories are tools which can be run at any time, from within any application. The next two menus, **File** and **Edit**, contain standard apple interface commands homogeneous to most applications. These commands will be discussed later in this chapter. The two other menus, **Label** and **Special**, are available only in the finder mode of operation. **Label** allows the user to change the appearance of icons on the desktop. The **Special** menu is used to turn off the computer, "empty the trash", and clean up the desktop. Detailed information regarding windows is located in reference pp 14–18 Macref.

2.1.7 Using An Apple Menu Item

Move the pointer to the menu and hold down the mouse button. With the mouse button pressed, drag the pointer to one of the Apple menu items, the Calculator, and release the mouse button. The calculator should appear, and quick calculations using either the mouse or the keyboard can be performed. The calculator is one of several desk accessories in the apple menu items menu. These desk accessories can be thought of as mini applications. They will run concurrently with other applications, and they require very little RAM (random access memory) to operate. Try this with several desk accessories until you are familiar with the use of them. For more information on Apple menu items, refer to pp 69, 106, and 216 Macref.

2.1.8 Balloon Help

Balloon help is a useful tool for the novice. This tool, located at the top right portion of the desktop, can be activated at any time by selecting **Show Balloons** from the **Balloon** menu, as seen in Figure 2.5. Try turning it on, and moving around the desktop as discussed

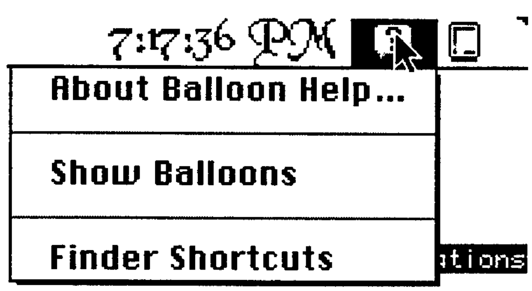


Figure 2.5: Accessing Balloon Help

in the previous sections. Notice that when you move the pointer over certain "hot spots", a help balloon pops up and describes the functionality of the area.

Some of the newest applications (Microsoft Word 5.0, for example) have built in balloon help. You certainly do not want to keep this help on all the time though, since it slows down the performance of the computer considerably.

2.1.9 Launching an Application

Applications may be launched several ways. One way is to select the application icon or file icon to launch (click once on the application icon) then choose **Open** from the **File** menu. An easier method to launch an application is to double-click on the icon. An application need not be launched directly, however. The user can also double-click on a file *created by* the parent application, which will launch the parent application and load the corresponding file into memory.

2.2 RUNNING APPLICATIONS

In this section, you will learn the basics of the TeachText text editor, creating and modifying files, and the *cut* and *paste* operations. Also, you will discover how to move about the workstation from within applications, editing and saving documents to your own user directory. A more formal discussion is presented in reference [2, Chapter 3].

2.2.1 Using Teachtext

Teachtext is a rudimentary word processor. Follow these steps to learn the basics of this application.

1. Find the application Teachtext. It should be located inside the folder called **Word Processing**. To open the **Word Processing** folder, position the mouse over the folder and double-click as shown in section 2.1.3.

2. Double-click on the Teachtext icon to launch. An untitled document window will automatically be created, since the application was launched. A blinking vertical bar called the insertion bar will be located in the top left corner of the Untitled window.

3. Start typing some text now, for example:

   `The quick brown fox jumps over the lazy dog.`

4. Save this newly created document by choosing **Save As...** from the **File** menu.

5. A dialog box will appear, as shown in Figure 2.6. Type in the name Test1, as shown in Figure 2.7, then click on the **Save** button.

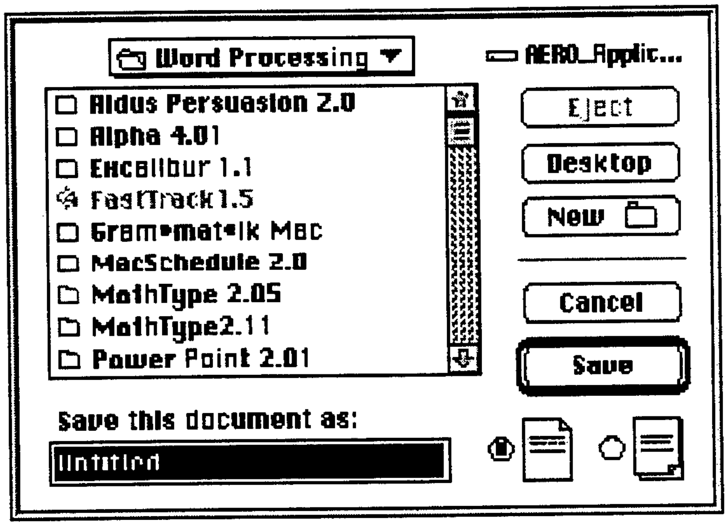


Figure 2.6: Dialog box asking the user how to save the newly created document

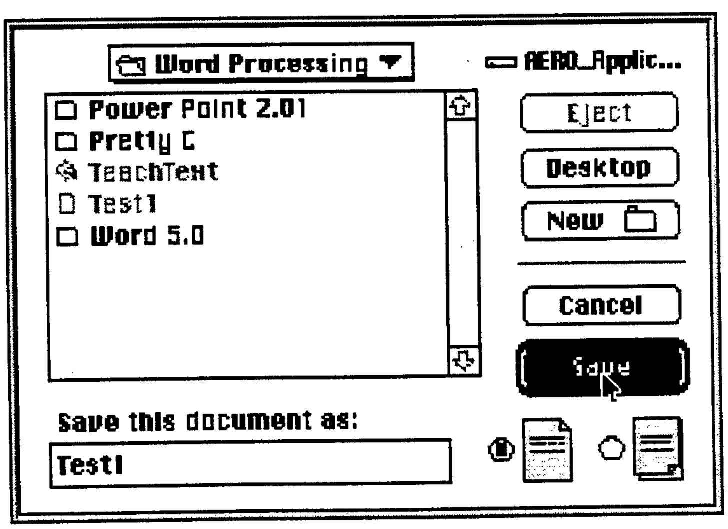


Figure 2.7: Saving the document as Test1

6. Now quit the program by choosing the **Quit** command from the **File** menu. A new file, called **Test1** will appear in the **Word Processing** folder as shown in Figure 2.8.

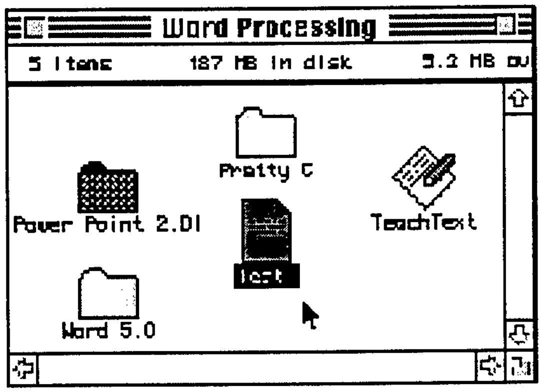


Figure 2.8: Word Processing window now contains a new file, called **Test1**

This file is located in the same folder as the parent application **TeachText**. Proper operating procedures call for each user to have a personal work area. You could create one now choosing the **New Folder** command in the **File** menu. Instead, however, you will create a new folder from within the application (described in the next section).

2.2.2 The Clipboard

The clipboard is an inherent part of the Macintosh operating system. The clipboard is a non-volatile portion of the memory of the Macintosh where text and pictures can be stored. The clipboard is accessed by *cutting, copying,* and *pasting.* These functions are demonstrated in the following steps:

1. Launch Teachtext by double-clicking directly on the newly created file, **Test1**.

2. Using the mouse, select a portion of the text. This operation is performed by placing the mouse pointer in the vicinity of the text to be selected, then using the click-and-drag technique described earlier. When a portion of text has been selected, the text will appear inverted (or highlighted) as shown in Figure 2.9.

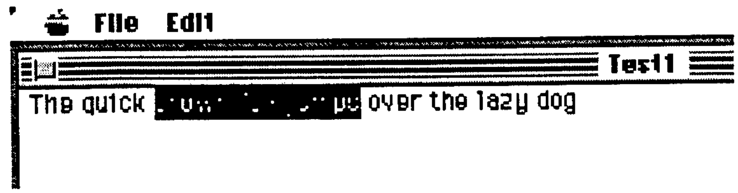


Figure 2.9: Text inverted by pressing the mouse and dragging along text

3. Release the mouse button and select **Cut** from the **Edit** menu. The text is removed. Actually, the text is stored in what is called the *clipboard.*

4. Now place the insertion bar somewhere else in the text, and select **Paste** from the **Edit** menu. Notice that the contents of the clipboard are placed at the insertion bar.

5. Select Paste again. Note that the contents of the clipboard have not changed, and the text is again inserted into the document. In fact, this text will remain there until replaced.

6. Now select another portion of text. This time choose **Copy** from the **Edit** menu. The contents of the clipboard have been replaced by the new selection, but the text was *not* cut from the document.

7. Place the insertion bar at the beginning of the document, and choose **Paste** from the **Edit** menu. The contents of the clipboard are now placed at the beginning of the document.

8. Now quit the program by choosing **Quit** from the **File** menu. Since you have made changes to the document, but have not saved the changes, the application knows to interactively ask whether or not to save the changes just made, as shown in Figure 2.10. Choose **Cancel** from the dialog box. This would save the changes to your original file. Instead, select **Save As** from the **File** menu.

9. The same dialog box appears as the first time you saved the document. Now you will create a new user folder. To do this, press and hold the mouse button down over the text that says "Word Processing". A pop-up menu will appear as shown in Figure 2.11. Drag the pointer down the list until the "desktop" is highlighted. Drag the pointer down the list until the "desktop" is inverted.

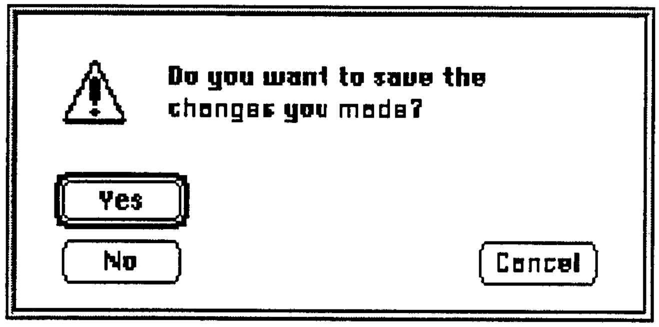


Figure 2.10: Program asks user if he wants to save changes

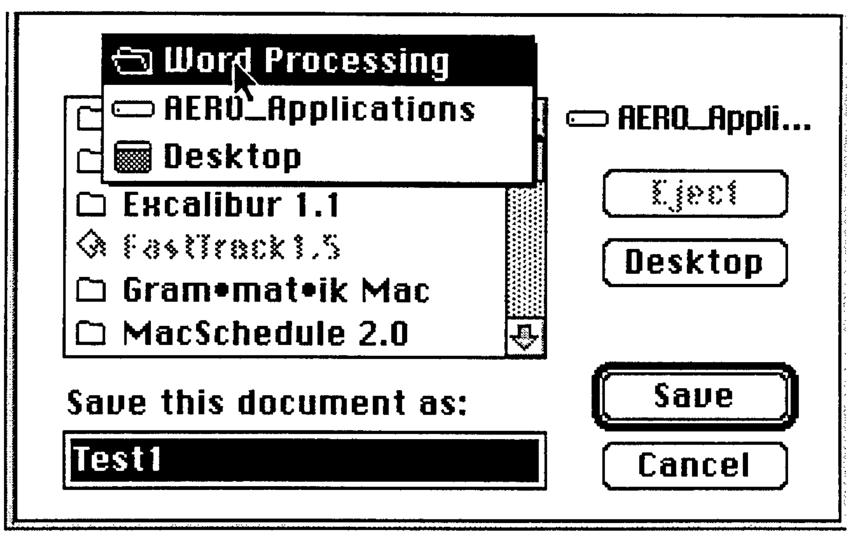


Figure 2.11: Switching Directories from within an application

10. You are now at the top level (or "root" directory) of the desktop. If you select **Save** now, you will save the file directly on the desktop. Since the point is to make your own user area, double-click on the text labelled "AERO_Users" now. This will navigate you into that hard disk.

11. Now press the button called **New Folder** in the dialog box. A dialog box asking you to name your new folder will appear.

12. Give your user folder a meaningful name (We called ours "User_Kalim", and will refer to the working directory as such from now on.

13. Click on the **Create** button to create the folder; then click on the **Save** button to save your data file to your newly created workspace.

14. Now quit the program once again. Your file should be located in the Hard Disk called **AERO_Users**, in a folder that you have just created. Find the file now by double–clicking to open the appropriate folder.

15. As a last exercise, restart the application by double-clicking on the **Test1** icon.

16. Again choose **Paste** from the **Edit** menu. The contents of the clipboard are again pasted into the document. Note that the contents remained even after the application ended. These contents can be pasted in other applications. In fact, even turning off the computer does not clear the contents of the clipboard.

2.2.3 Switching Between Applications

The Macintosh is a multitasking environment which allows several applications to run concurrently, or in other words, perform background processing. In the upper right corner of the desktop, a miniature icon shown in Figure 2.12 can be seen.

A number of applications can be running concurrently. The amount of RAM (Random Access Memory) and application sizes determine the maximum number of applications the user can run simultaneously. To see how much RAM is available, choose **About this Macintosh...** from the **Apple** menu when in the finder (you cannot do this from within an application, because each application customizes this space for its own logo. To demonstrate the idea of multitasking, follow the steps described below:

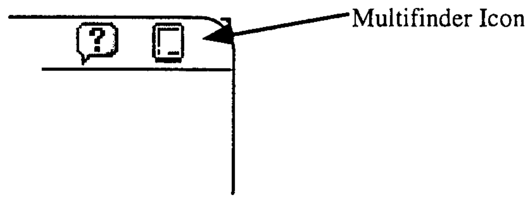


Figure 2.12: Miniature icon showing what applications are running

1. Launch the application Teachtext once again. Notice that a miniaturized version of the Teachtext icon appears in the top right portion of the monitor.

2. Now position the pointer on the multifinder icon, click and hold, and select **Finder**. This will change the foreground application to the finder. Also note that the icon changes from that of the application to the Finder icon. From here the user can launch another application without disrupting Teachtext, which is already in use.

3. From the Finder, launch **PowerPoint**. While the application is starting, the small icon in the top right corner of the desktop will change to a miniature **PowerPoint** icon. The window will change to generic **PowerPoint** untitled window.

4. Switch from **PowerPoint** to **TeachText** by pressing the mouse button over the miniature **PowerPoint** icon and selecting **Teachtext** from the list of menu items.

5. The cut/paste exercise we demonstrated in Section 2.2.2 has full functionality between applications. Type some text in the **TeachText** untitled window as before.

6. Select the text by dragging the mouse across the text, and copy it by choosing **Copy** from the **Edit** menu.

7. Switch applications as in step 2, only this time, switch to the **PowerPoint** program.

8. Select **Paste** from the **Edit** menu. Note that the text has been pasted into the new document. Pictures and tables can also be pasted in this fashion.

Working with several applications simultaneously is explained in greater detail in reference [2, pp 72–78].

2.2.4 Trash Can

This section will show you how delete an item from the hard disk (*never delete an item that is not yours*).

1. Go to your user folder on the **AERO_Users** hard disk.

2. Find the **Test1** file, and drag it into the trash can. You will know when it is actually in the trash can, and not just on top of it because the trash can will become highlighted.

3. Now release the mouse. The trash can should "expand" as though it were full. This means the trash is not empty. You can recover the data from here simply by clicking on the trash can and dragging your file out of the trash.

4. To empty the trash (non-recoverable) select **Empty Trash** from the **Special** menu.

2.3 Summary

This chapter was designed to familiarize an individual with the Macintosh. These basic concepts should now be familiar. The user should be able to:

- use the mouse
- select items from the menus
- use desk accessories
- launch applications/documents
- cut/copy/paste
- trash

Other topics that have not been discussed here but will be of interest as you become more familiar with the operating system are aliases, stationary pads, file finding, and file sharing, to name a few. These are not considered essential but are very useful and are found once again in reference [2].

Chapter 3

4th DIMENSION Fundamentals

Databases are a very important tool in data analysis. The tools provided by a database allow us to manipulate, search through, and sort data to present the data in a format that is useful to the analyst. To be functional, a database must fulfill these requirements, and also be relatively straightforward in its implementation.

This chapter is presented to acquaint the user with the use of 4th DIMENSION, a commercial database package developed by ACIUS software. After sampling many databases on various platforms, 4th DIMENSION was chosen for its flexibility and ease of use. Also, the program has the ability to import graphics, which may be included in future releases of the system effects database.

The discussion presented here is not comprehensive, but should serve as a guideline for a novice to familiarize him with the application. More complete description can be found in references [5] and [6].

AIRBASE will be used to demonstrate the basic functions of the 4th DIMENSION application, but no knowledge of the AIRBASE concept is necessary for this demonstration. See Chapter 5 for a full description of the AIRBASE structure. To begin, double-click on the file **Airbase** or the file **Airbase.data**, both located in the **Airbase** directory on the **AERO_DB** volume. After launching the database, the environment you will be placed in will either be the **User** environment (see Figure 3.1), or the **Design** environment (see Figure 3.2). When the application is launched, if you are in the **Design** environment, switch to the **user** environment by selecting **User** from the **Use** menu.

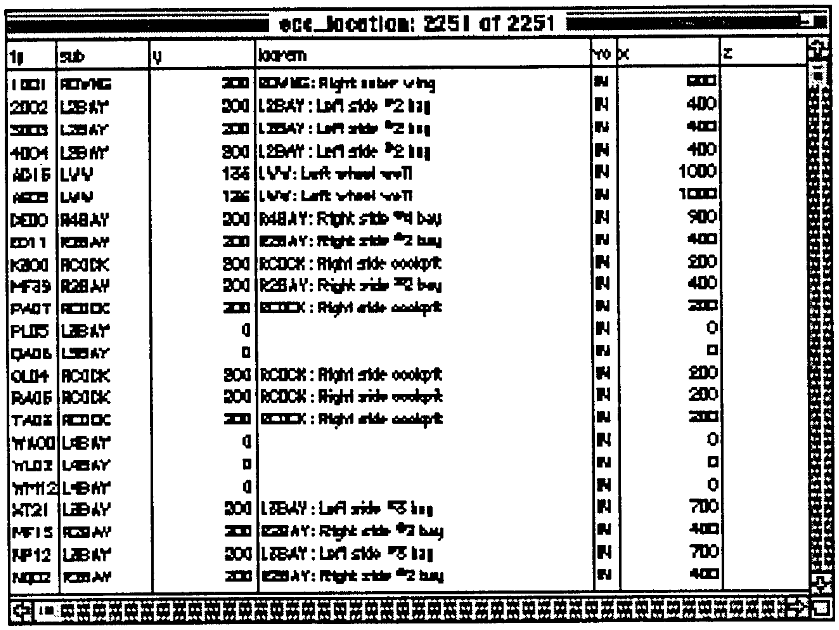


Figure 3.1: User Environment

3.1 User Environment Overview

The function of the User environment is to allow you to manipulate and view data. A complete set of tools allow the user to comprehensively search through and sort data, modify existing data sets (although AIRBASE philosophy stresses that the data should *not* be contaminated via user modifications), import and export data from/to other environments, and create reports. The user environment (as shown in Figure 3.1) contains the following menus:

- File
- Edit
- Use
- Enter
- Select
- Report

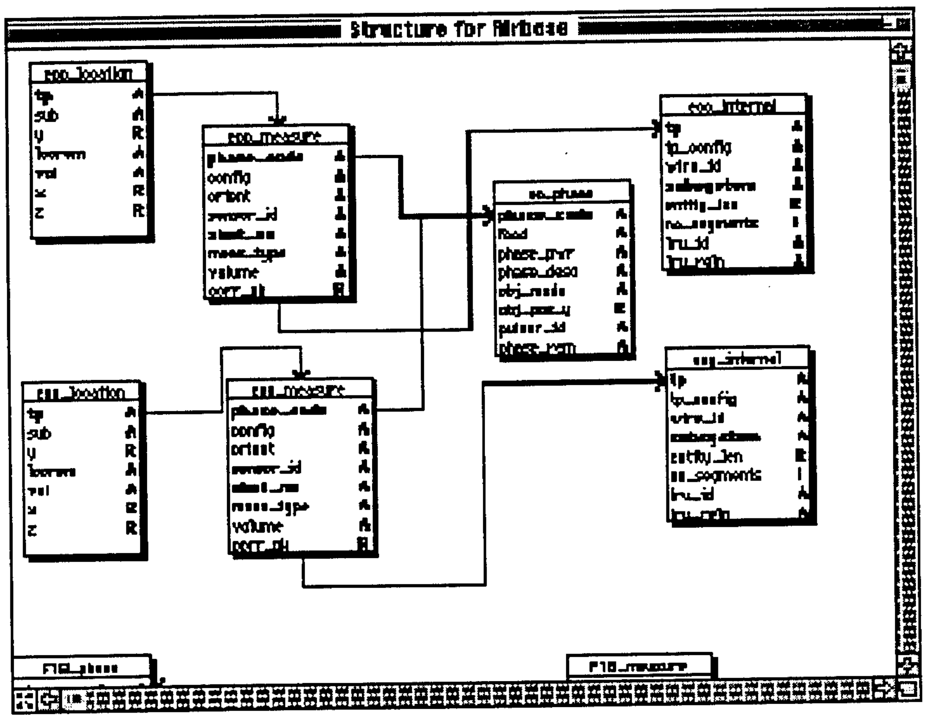


Figure 3.2: Design Environment

- Special

These menus can be customized, but that is beyond the scope of this text. Briefly, the aforementioned items will be discussed.

3.1.1 File Menu

The file menu contains all of the items necessary for file operations. These operations include: creating a new database, opening a pre-existing database, importing and exporting data in various formats, and changing the file or layout.

3.1.2 Edit Menu

The Edit menu contains the standard Macintosh editing commands that allow the user to cut/copy/paste, as well as other standard editing functions.

3.1.3 Use Menu

The Use menu allows the user to change between working environments. The environments in 4th DIMENSION are Design (see Section 3.2), User, and Runtime (a custom environment not implemented on AIRBASE).

3.1.4 Enter Menu

The Enter menu allows the user to enter and modify data. Again, users of AIRBASE should not perform these actions on the populated database, as this may lead to data corruption.

3.1.5 Select Menu

The Select menu allows the user to search through and sort data. In the analysis process, this menu will be very useful.

3.1.6 Report Menu

The Report menu contains menu items to create quick reports on the fly, generate labels, and perform simple graphing.

3.1.7 Special Menu

The Special menu contains items that allows users to run user-created procedures (*very important*), and edit ASCII mapping (not pertinent to AIRBASE users).

3.2 Design Environment Overview

The Design environment allows the user to create and modify a database structure, design layouts for viewing in the User environment, and create complex reports. A full-featured layout editor that functions as a complete drawing program is included. A programming language that allows the user to completely control data, as well as link external procedures written in other languages, is also included, but will not be covered in this overview, due to the nature and complexity of the topic. Instead, the user who is interested in programming 4th DIMENSION procedures is referred to reference [3, Chapter 6]. The design environment will look like Figure 3.2 when first entered. The menus found in the Design environment are as follows:

- File
- Edit
- Use
- Design
- Structure

The File, Edit, and Use menus will not be discussed here, as they are basically the same as those discussed in Section 3.1. The the Design and Structure menus will be described here.

3.2.1 Design Menu

The Design Menu contains virtually all of the useful items in the environment.

- **Structure** - The default selection for the design menu is the structure. The structure displays the images of the files in the database and graphically shows relationships between files. The structure window allows the user to create files and fields in the database, specify field types and attributes, relate and modify file types, and set access privileges.

- **Layout** - The layout editor allows the user to create and modify a layout for display in the User environment. More detail of the Layout editor will be discussed in Section 3.4.

- **Procedure** - The procedure editor allows the user to generate programming instructions that process the information contained in the database. The procedures can perform computations on files, transfer data between files, or control data as it is input. Procedures are particularly important when you want to create a stand-alone runtime application. Procedures are also relevant when you want to perform complex data manipulation, but are not essential for initial database queries, and therefore are not included in this basic tutorial. The design reference manual [3] is a comprehensive text that covers procedure generation. For AIRBASE, we have generated a necessary procedure and its *use* will be covered in Section 3.3.5.

- **Menus** - The 4th DIMENSION menu bars are all customizable. For example, if you created a custom environment, you could make menu items for the user to perform basic query functions, making data access even easier. Unfortunately, versatility is lost when limiting the user to only certain functions. For this reason, we chose not to implement a custom environment for AIRBASE. Reference [3, Chapter 7] describes how to implement custom menus.

- **Passwords** - Many levels of password protection are available in this program. They are not implemented in AIRBASE (yet) and will not be discussed in this document. For more information, we would refer you to [3, Chapter 8].

- **Lists** - A list is a set of possible values for a field. A list can provide the user with choices for data entry in a field, or perform quality control on data entry. Lists are not necessary for this application, as data is not manually entered into the database, but rather imported from a pre-existing spreadsheet. For more information, please refer to [3, Chapter 9].

3.3 User Environment – an Analyst's Perspective

This section will provide the user with the knowledge necessary to navigate the user environment, perform simple database queries, and export information. In 4th DIMENSION, information is stored in files. Each file stores various information on a subject. For example, the Phase files describe the tests that were performed for a specific aircraft. The F18 phase table contains all the different test configurations used in the F18 test. Each file contains a number of fields. For example, the measurement tables contain information on each measurement shot, such as the type of probe used in the measurement, the type of measurement (skin current, voltage, etc.), the peak of the waveform, etc. These rudimentary descriptions are not meant to suffice for the AIRBASE data, but rather as examples of the terms "file" and "field". For a detailed description on the exact contents of the AIRBASE database, refer to Chapter 5.

The terms *input* and *output* will be used throughout this section. An input layout displays only one record at a time, and the data can be modified in this mode. Output layouts are meant for display purposes, and as much data as can be viewed is displayed. AIRBASE users will not be entering data into the database manually, so the input mode is not often used.

3.3.1 Choosing Files and Layouts

First and foremost, the user must be able to successfully navigate through the various files and layouts available to him, this will be discussed presently. If the appropriate layout does not exist, the user must create his own, as discussed in Section 3.4.

When in the user environment, a file should appear. Changing files and layouts can be accomplished in two ways. The first method is as follows:

1. choose **Choose File/Layout...** from the the **File** menu. The dialog box shown in Figure 3.3 will appear.

2. Select a file by *double-clicking* a filename, located on the left side of the dialog box. 4th DIMENSION will then expand a list of the layouts that belong to the file. If you inadvertently clicked on the wrong file, you can compress it by double-clicking on the filename again. This acts as a toggle between expanded and normal view.

3. The current input and output layouts will be denoted by a symbol, either *I* for input, *O* for output, or *B* for both. To change a view, select the desired layout and press the appropriate button, **Input** or **Output** to make these the active view. On the right side, a miniature view of the layout will be displayed. If this is the desired layout scheme, click on the **Choose** button, or press the **RETURN** key.

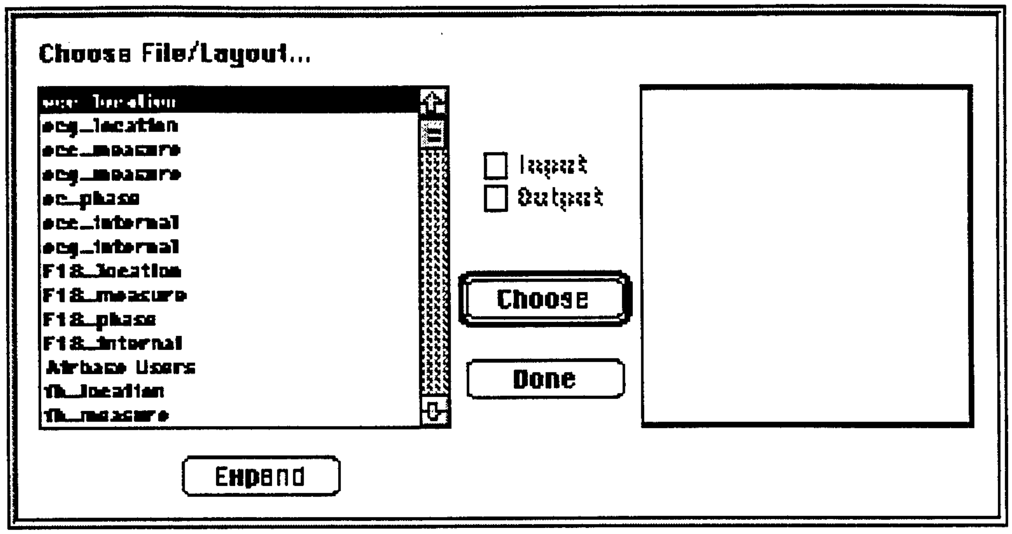


Figure 3.3: Dialog box of the **Choose File/Layout...** menu item.

The second method for modifying a file/layout is to use the "List of Files" window, as shown in Figure 3.4 To use this method, follow these steps:

1. If the "List of Files" window is not visible, hold down the Command key and press the Space bar, bringing it to the foreground.

2. To switch files, click on the appropriate file in the window.

3. To change to another layout, press the mouse on either the **O** or the **I** to the left of the file you are changing to, and after a few milliseconds, a pop-up menu appears. This menu displays the list of available layouts for the file. The underlined layout is the one presently selected. To change layouts, keep the button pressed while dragging the pointer up or down until the desired layout is inverted in appearance.

4. To hide the window after selecting, press the key sequence command-space again, or position the mouse over the file window, and click once to make that window active.

A more detailed description of files and layouts can be found in reference [5, Chapter 2].

3.3.2 Selecting Data

To select a subset of the whole file, follow these steps:

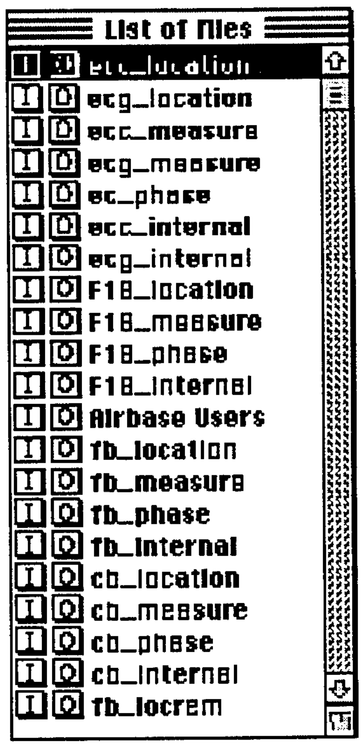


Figure 3.4: "List of Files" Window

1. Click on the first record you want to select.

2. To select a group of contiguous records beginning with the record you selected, hold down the Shift key and click on the last record you want to select.

3. If you want to select multiple non-contiguous records, hold down the Shift key and click on the records you want to select.

4. After you have selected all of the records of interest, choose **Show Subset** from the **Select** menu. This is called the *current selection.* To revert to the full display, choose **Show All** from the **Select** menu.

3.3.3 Searching for Data

Searching is one of the most common database operations. 4th DIMENSION provides several powerful tools to perform such searches. These include

- Search Editor
- Search by Layout
- Search and Modify
- Search by Formula
- Searching via 4th DIMENSION's procedural language

Only the **Search Editor** method will be discussed here. This is enough for all but the most complex search operations. For explanation of the other methods, see reference [5, Chapter 4].

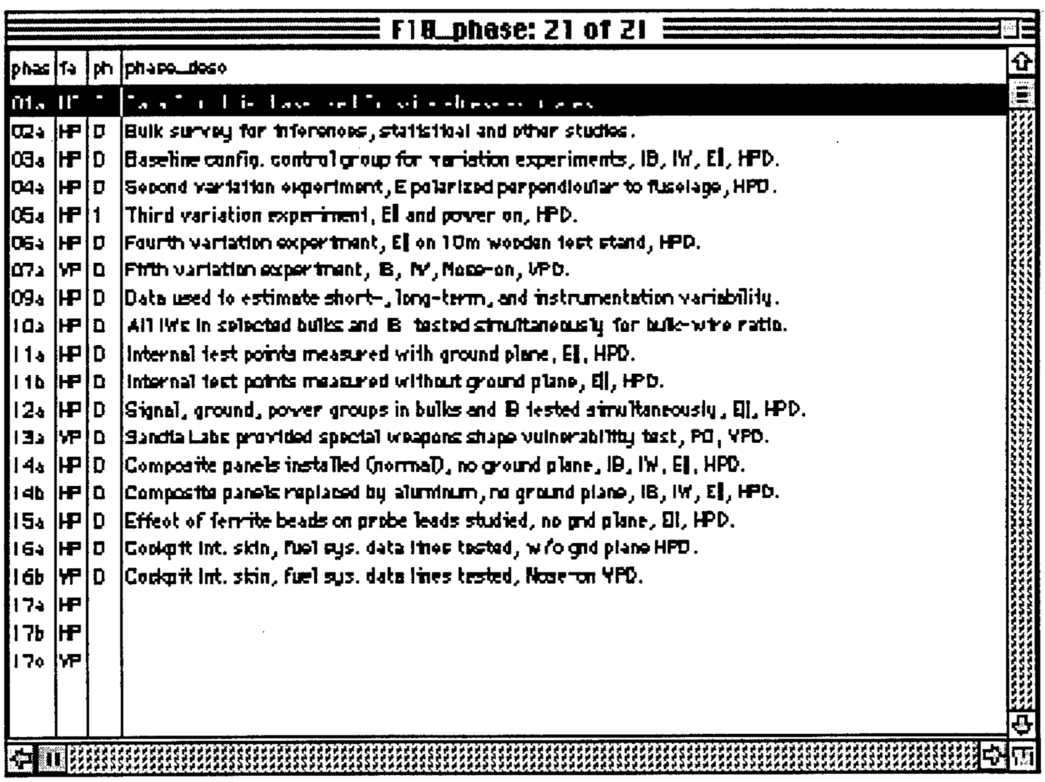
Database queries (searches) are executed when you want to access a specific subset of records from the file. For example, you might search a measurement file for all records taken during test phase **01a**. This is a subset of the entire file contents, called the *current selection.*

Any search is performed by specifying *search conditions.* Search conditions are a set of instructions that tell the database which records to include in the new subset (the current selection). A search condition will always have three elements: the field name, comparison operator, and a value to compare with. The speed of this search depends on both the amount of data contained in the file, and more importantly, whether or not the search field has been *indexed.* An *indexed field* is an ordered field, which increases the amount of storage space required, but allows much faster access to the data contained therein.

We indexed the fields that are most important in database queries, and these fields are easily distinguished from non-indexed fields, as shown in the following search example. This simple example searches the **f18_measure** file for all records.

1. Verify you are in the user environment (select **User** from the **Use** menu).

2. First, we will investigate what **phase_code = 01a** really means. Switch to file **f18_phase**. Recall that switching tables is described in Section 3.3.1. In the phase description field, you will notice that **phase_code = 01a** means "Data from this phase used for wire stress estimates". See Figure 3.5 for the layout.



F18_phase: 21 of 21

phas	fa	ph	phase_desc
01a	[illegible]	[illegible]	[illegible]
02a	HP	D	Bulk survey for inferences, statistical and other studies.
03a	HP	D	Baseline config. control group for variation experiments, IB, IW, E‖, HPD.
04a	HP	D	Second variation experiment, E polarized perpendicular to fuselage, HPD.
05a	HP	1	Third variation experiment, E‖ and power on, HPD.
06a	HP	D	Fourth variation experiment, E‖ on 10m wooden test stand, HPD.
07a	VP	D	Fifth variation experiment, IB, IW, Nose-on, VPD.
09a	HP	D	Data used to estimate short-, long-term, and instrumentation variability.
10a	HP	D	All IWs in selected bulks and IB tested simultaneously for bulk-wire ratio.
11a	HP	D	Internal test points measured with ground plane, E‖, HPD.
11b	HP	D	Internal test points measured without ground plane, E‖, HPD.
12a	HP	D	Signal, ground, power groups in bulks and IB tested simultaneously, E‖, HPD.
13a	VP	D	Sandia Labs provided special weapons shape vulnerability test, PO, VPD.
14a	HP	D	Composite panels installed (normal), no ground plane, IB, IW, E‖, HPD.
14b	HP	D	Composite panels replaced by aluminum, no ground plane, IB, IW, E‖, HPD.
15a	HP	D	Effect of ferrite beads on probe leads studied, no gnd plane, E‖, HPD.
16a	HP	D	Cockpit int. skin, fuel sys. data lines tested, w/o gnd plane HPD.
16b	VP	D	Cockpit int. skin, fuel sys. data lines tested, Nose-on VPD.
17a	HP		
17b	HP		
17c	VP		

Figure 3.5: F18 phase table, phase code = 01a is highlighted

3. Now switch to file **f18_measure** and choose the layout called **all fields out**. This translates to "show all fields in this file".

4. Choose **Search Editor...** from the **Select** menu. A dialog box similar to that shown in Figure 3.6 will appear.

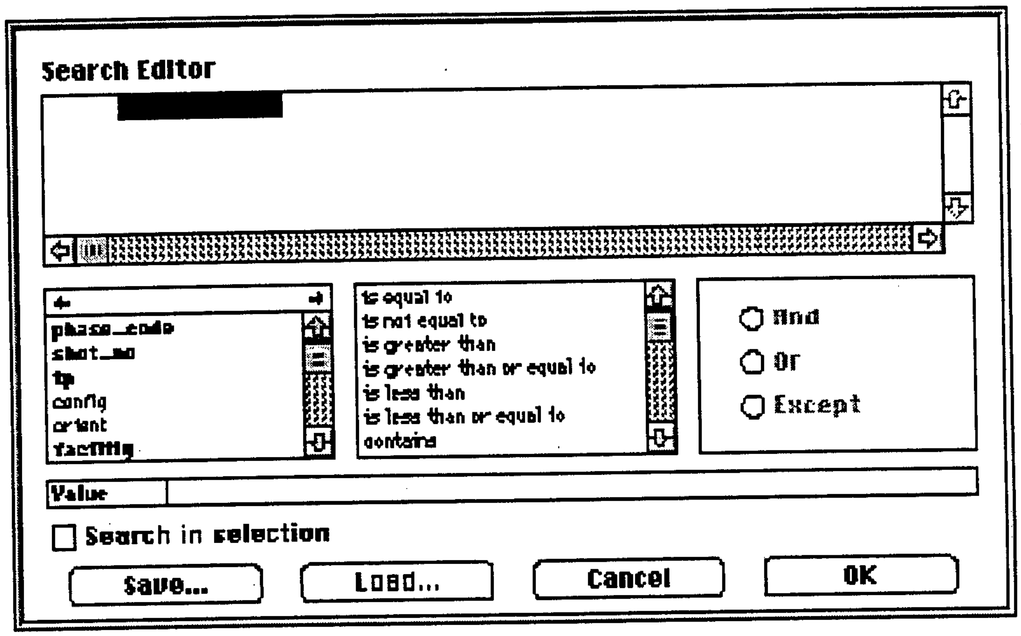


Figure 3.6: Search Editor dialog box for f18 measurement file

5. All the fields in the **f18_measure** table are listed in the left column. *All indexed fields are in bold-type (phase_code, tp, and shot_no are examples).* You will now generate a simple data search that finds all records in the file that have a phase_code equal to 01a. First, choose **phase_code** from the left column (note that this is an indexed field, so the search will be very fast). Next, choose **is equal to** from the comparison column (middle column). Now in the **Value** area, type in the sequence **01a** (note that the database search is *not* case sensitive, so **01A** will also suffice). Finally, press the **OK** button, to start the search. A screen similar to Figure 3.7 should appear.

To perform complex queries, simply combine sets of conditions with the "And", "Or", and "Except" qualifiers. Another way to generate a complex search is to perform a simple search resulting in a subset of all the records. Then perform another simple search with the **Search in selection** box checked.

F18_measure: 1009 of 2412

phas	shot_	tp	conf	or1	fa	me	meas	vo	senso	se	integr	in1	data1	filter	signal	signal	signal	signal	nh
01a	1467	AA34			HP	IV	AMPS	N	C140	I		0	D404	F101					3
01a	1468	AA34			HP	IV	AMPS	N	C140	I		0	D404	F101					3
01a	0730	AE02			HP	IV	AMPS	N	C147	I		0	D411	F101					4
01a	9894	AL18			HP	IV	AMPS	N	C146	I		0	D402	F101					3
01a	9865	AL19			HP	IV	AMPS	N	C146	I		0	D402	F101					3
01a	9867	AL25			HP	IV	AMPS	N	C152	I		0	D402	F101					3
01a	9909	AL24			HP	IV	AMPS	N	C146	I		0	D402	F101					3
01a	9869	AL25			HP	IV	AMPS	N	C147	I		0	D402	F101					3
01a	0032	AL26			HP	IV	AMPS	N	C152	I		0	D402	F101					3
01a	0254	AL26			HP	IV	AMPS	N	C147	I		0	D411	F101					3
01a	0300	AL28			HP	IV	AMPS	N	C108	I		0	D411	F101					3
01a	0225	AL30			HP	IV	AMPS	N	C152	I		0	D411	F101					3
01a	0100	AL31			HP	IV	AMPS	N	C152	I		0	D404	F101					3
01a	9747	AL31			HP	IV	AMPS	N	C145	I		0	D404	F101					3
01a	9749	AL31			HP	IV	AMPS	N	C145	I		0	D404	F101					3
01a	9822	AL31			HP	IV	AMPS	N	C145	I		0	D404	F101					3
01a	9912	AL31			HP	IV	AMPS	N	C152	I		0	D402	F101					3
01a	9934	AL31			HP	IV	AMPS	N	C152	I		0	D402	F101					3
01a	0169	AL32			HP	IV	AMPS	N	C152	I		0	D411	F101					3
01a	0176	AL33			HP	IV	AMPS	N	C147	I		0	D411	F101					3
01a	0298	AL34			HP	IV	AMPS	N	C147	I		0	D411	F101					3
01a	0448	AL35			HP	IV	AMPS	N	C147	I		0	D411	F101					3
01a	0258	AL36			HP	IV	AMPS	N	C152	I		0	D411	F101					3

Figure 3.7: Selection of records in f18 measurement file with test phase code = **01a**

3.3.4 Sorting Data

You may want to view data in a particular order, so sorting the data is recommended. Sorting reorders data according to the values in the database. You may sort records on as many as 30 fields. You can specify ascending or descending order for each field selected, and you are allowed to sort by alphanumeric characters or numbers. This can be accomplished in several ways. In the user environment, you can use the **Sort Selection...** command or the **Sort File...** command. From the design environment, you can put sorting criteria in procedures (for more information on this method of sorting, consult reference [3, Chapter 6] on creating procedures).

You can perform a temporary sort on the current selection by choosing **Sort Selection...** from the **Select** menu. This method of sorting is preferable to the alternative, **Sort File...** because the latter *permanently* sorts the contents of the entire file.

As an example, suppose we want to perform a sort on the selection generated in Section 3.3.3 in order of ascending **Shot Number** and **Test Point**. To sort the data follow these steps:

1. Verify the current selection is the same as in Section 3.3.3. The selection should resemble Figure 3.7.

2. Choose **Sort Selection...** from the **Select** menu. The sort editor, shown in Figure 3.8, will appear.

3. Notice that the left column contains all the fields of the measurement table. Choose **shot_no** and then **tp** from this list. This function performs the sort based primarily on the shot number, and then the test point.

4. The default is an *ascending* sort, but this is easily changed by clicking the sort arrows to the right of the screen. Click on the arrow to the right of the **tp** sort parameter. The sort editor window should now look like Figure 3.9

The previous steps are the same as the **Sort File...** command, but we advise against using the latter method of sorting, because it *permanently* modifies the data order.

3.3.5 Executing Procedures – Decompressall

AIRBASE waveforms are stored on the database in a compressed format. This data requires a procedure called **Decompressall** for decompression. The actual compression algorithm is

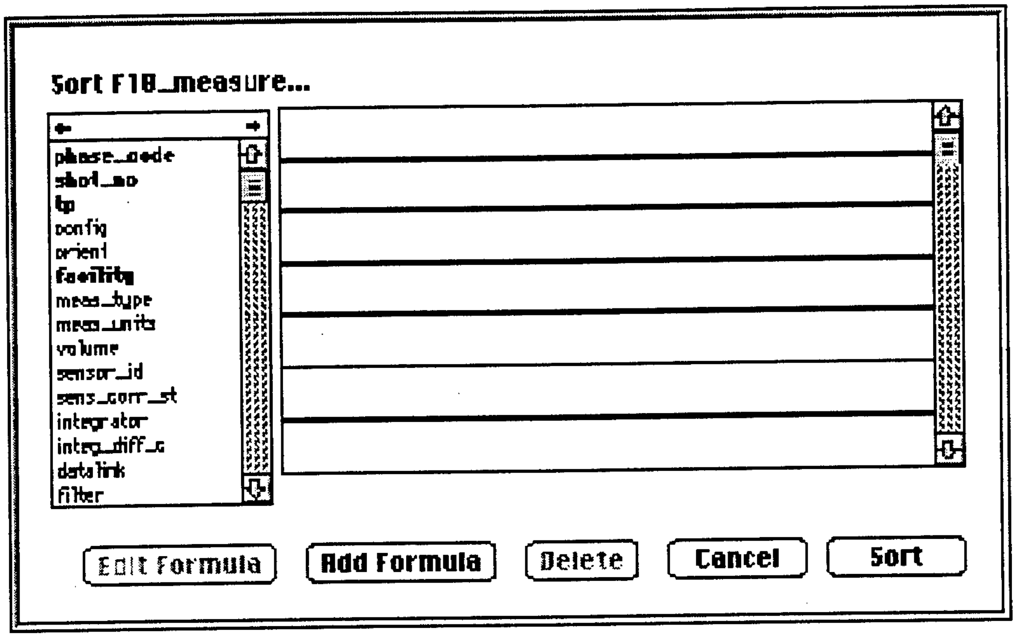


Figure 3.8: Sort Editor for f18 measurement table

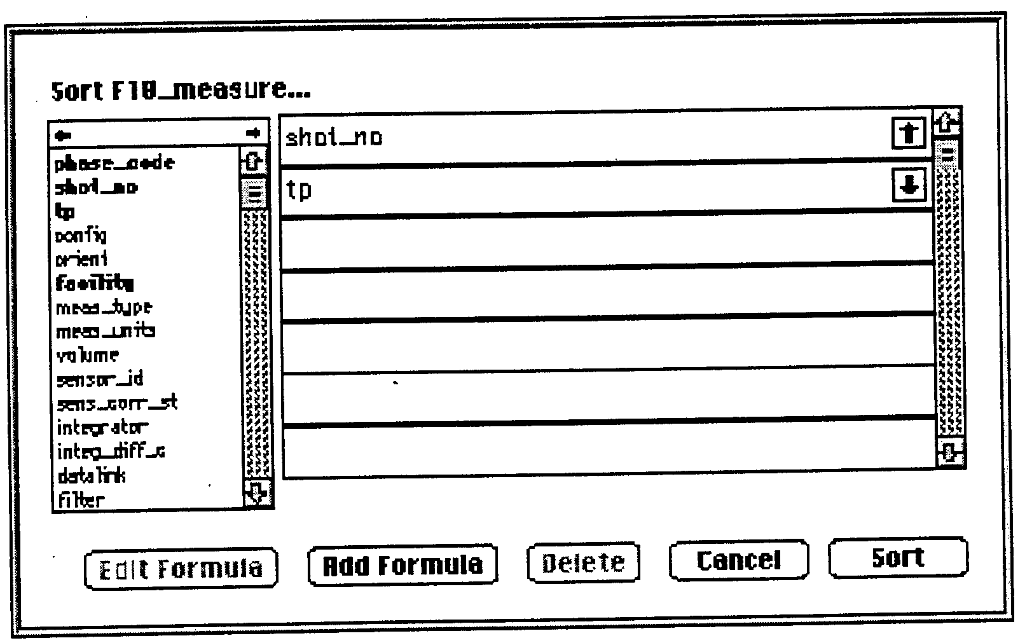


Figure 3.9: Sort shot number – ascending ; test point – descending

discussed in detail in reference [1, Appendix E].

Executing a procedure is similar to running a generic program in any programming language. The procedures may vary so greatly that all procedures must be considered independently. This section describes the method used to execute a generic procedure, and specifically the procedure **Decompressall**.

Decompressall is a procedure that decompresses waveform data in the current selection, exporting it to the hard disk in a format specified by the user. The waveforms are located *only* in the measurement files, so it is pointless to run the procedure if you are in any other file (phase, internal, or location files). This procedure is the *only* one that a user need to know. The other procedures contained in AIRBASE were created with specific analysis tasks in mind. These procedures were included in the deliverable version of AIRBASE only to give the user some examples of how to write procedures, and have no general-purpose value.

In this example, you will decompress a few waveforms from the f18 measurement table. These waveforms will be decompressed into "MATLAB-format". A binary data format directly readable by the signal processing package, MATLAB. This example of **Decompressall** is outlined in the following steps:

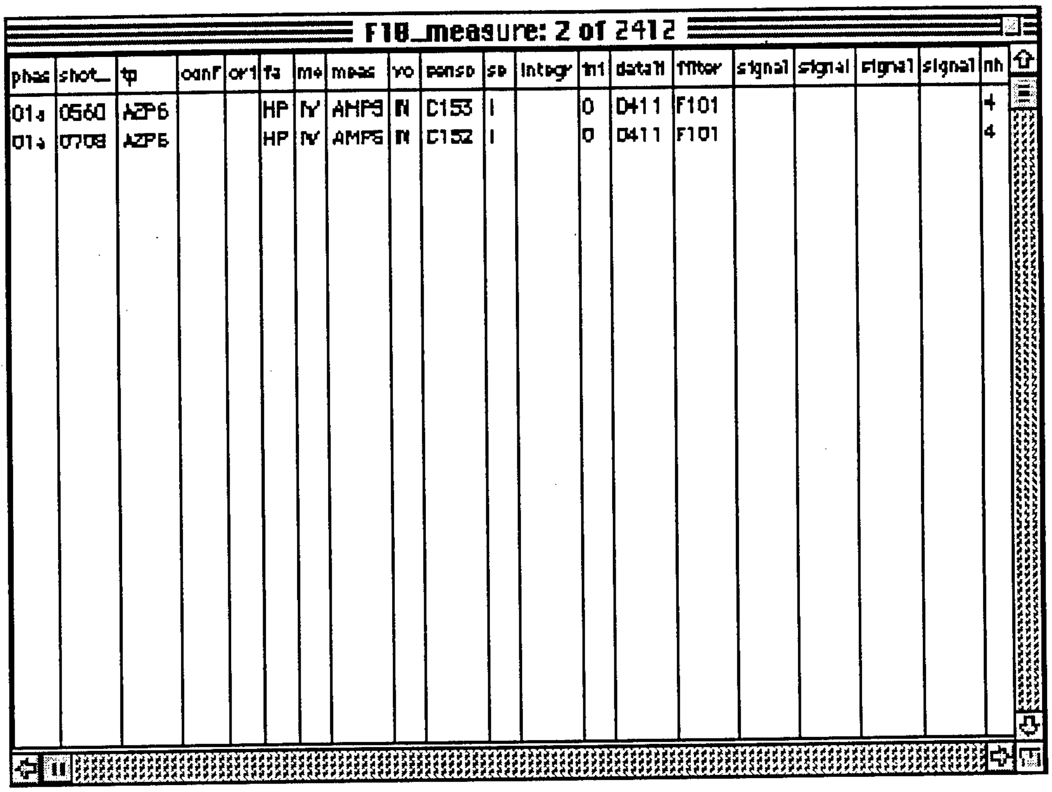
1. Switch to the f18 measurement table. Select **tp = AZP6** from the search editor (details of searching are described in Section 3.3.3). Two records should be listed after the search, as shown in Figure 3.10.

2. Select **Execute Procedure**... from the **Special** menu. 4th DIMENSION displays the "Execute procedure..." dialog box as shown in Figure 3.11.

3. Select the procedure **Decompressall** from the scrollable list. Click execute (or you may choose to simply double-click on the name of the procedure.)

4. This procedure requires a few steps be taken during execution. The first dialog box will look like Figure 3.12. You *must* type the name of the file in which you are currently working. This tells the database from which file to decompress the waveforms.

5. Next, a dialog box asking you where to store the data will appear. The default location (no entry) is the database directory itself. To select a different directory, type the *full* pathname (similar to DOS or UNIX, except the directory delimiter is ":" instead of a "\" or a "/". For example, if you want to put it on the AERO_Users hard disk, in a folder called "User_Kalim", you would type "AERO_Users:User_Kalim:". If you have



F18_measure: 2 of 2412

phas	shot_	tp	conf	ori	fa	me	meas	vo	senso	se	integr	ini	datafi	filter	signal	signal	signal	signal	nh
01a	0560	AZP6			HP	IV	AMPS	IN	C153	I		0	D411	F101					4
01a	0708	AZP6			HP	IV	AMPS	IN	C152	I		0	D411	F101					4

Figure 3.10: Result of search for all records with test point equal to AZP6

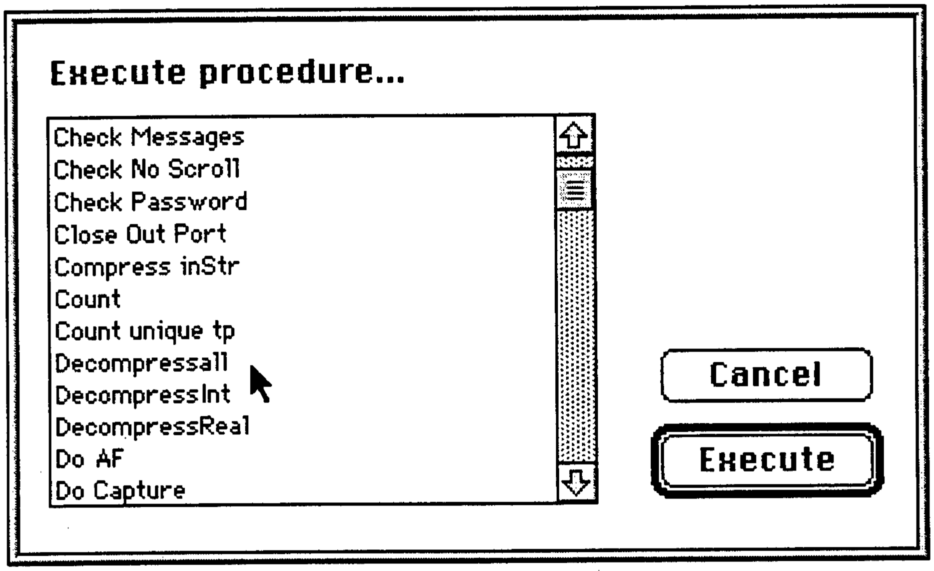


Figure 3.11: Execute Procedure Dialog Box

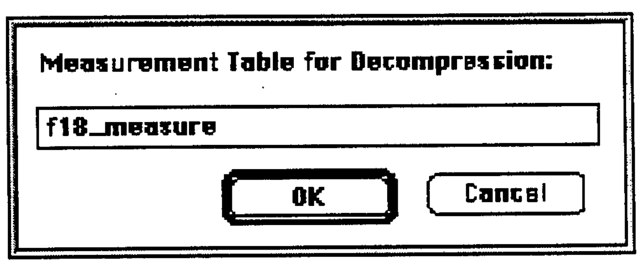


Figure 3.12: First dialog box of **Decompressall**

a personal directory on the workstation, type that path in now. In Figure 3.13, my path was chosen.

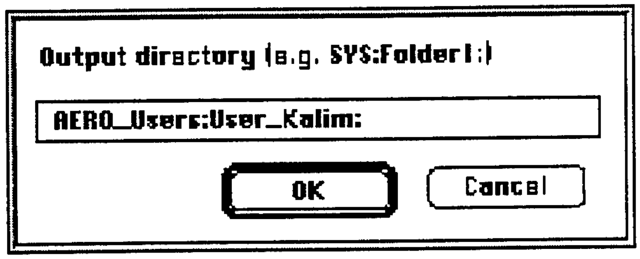


Figure 3.13: Second dialog box of **Decompressall**, choosing my home directory

6. Finally, a decompression format must be chosen. The four options shown in Table 3.1 are currently available. The dialog box is the same as shown in Figure 3.14.

Table 3.1: Data Decompression File Formats

Option	Data Format
1	header information and waveform in ASCII X-Y pairs
2	header information and Autoregressive Coefficients
3	header and AR coefficients and data in XY pairs
4	Waveform in binary format compatible with MATLAB

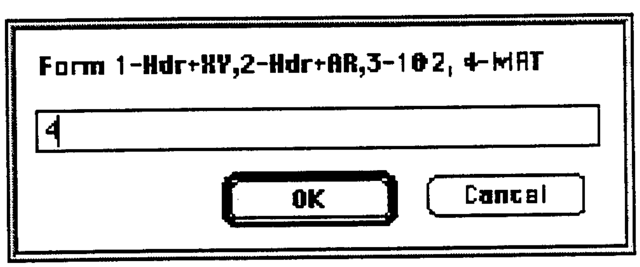


Figure 3.14: Third dialog box of **Decompressall**, selecting decompression format.

Enter "4" in the dialog box. The cursor will be replaced by a spinning wheel while the decompression is taking place. Since only 2 waveforms are being decompressed, the delay will hardly be noticeable. Waveform decompression only takes approximately 1.5 seconds per data file.

7. Now switch to the finder. In whatever folder you chose for decompression (default is the AIRBASE folder), you will find two newly created files (if you are in "View by Icon" mode, the folder should look similar to Figure 3.15). We have fooled these files into thinking they were generated by MATLAB, so double-clicking on either of them has the effect of launching MATLAB and loading the data into MATLAB's internal work area. We will discuss MATLAB shortly (in Chapter 4) and you will perform some basic manipulations with these files, so *don't forget where you put them!*

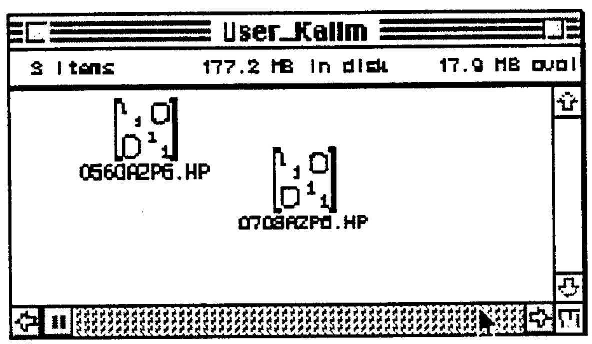


Figure 3.15: Newly created files in the User_Kalim folder

3.3.6 Exporting Data

The import/export utility provided by 4th DIMENSION provides an easy method of transferring data to and from the database. In AIRBASE, we assume the user will not be importing any data, so only data exportation is discussed. Importing data is very similar, so these guidelines should be enough to perform both operations, *but we must emphasize that data should only be imported to another database of your own design, not into AIRBASE proper.* If more detailed information is required for your specific importing/exporting needs, refer to [5, Chapter 12].

Data can be exported from 4th DIMENSION so that it can be used in other applications such as spreadsheets, graphics packages, and word processors. Data can be exported in different file formats, such as

- **SYLK** – SYymbolic LinK format. This format is used by some databases in the DOS world.
- **DIF** – Data Interchange Format. This format is readable by such spreadsheets as Excel, LOTUS-123, etc.
- **Text** – Text format is readable by most of today's spreadsheets, word processors, and graphics packages. It simply contains the ASCII data separated by field delimiters and record delimiters. The default field delimiter is ASCII code 9 (the "tab" character), and the default record delimiter is ASCII code 13 (the "carriage return" character). The user can change these to whatever delimiters he chooses, but for most packages this is not necessary.

In this example, we will export all of the shot numbers, test points, and corresponding corrected peaks contained in the **F18 Measurement** file. To perform this task, follow these procedures:

1. Select **f18_measure** from the list of files or by choosing it from **Choose File/Layout...** in the the **File** menu.
2. Select **Export Data...** from the **File** menu. The Export Data dialog box is shown in Figure 3.16.
3. Enter a name for the file in the "Save As" area (I named the file "exported data from f18"). Change the directory to your user area, if you have created one. Otherwise, save the data in the AIRBASE area.
4. Select a file format for the data to be exported (**Text** format will suffice, which should be the default format).
5. Click on the appropriate field names displayed in the bottom left corner of the dialog box. For this example, select **shot_no**, and click on the "Append" button. Next, click on **tp** and then "Append". Finally, scroll down the list until you find **corr_pk**, click on it to select it, and then click on the "Append" button once again. The dialog box should now look something like Figure 3.17.

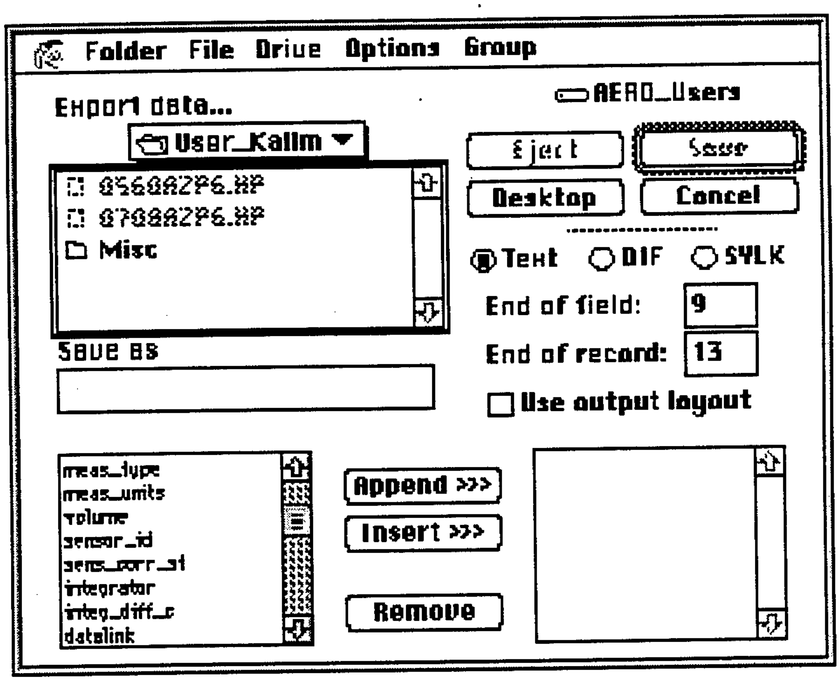


Figure 3.16: Export Data dialog box for F18 measurement table

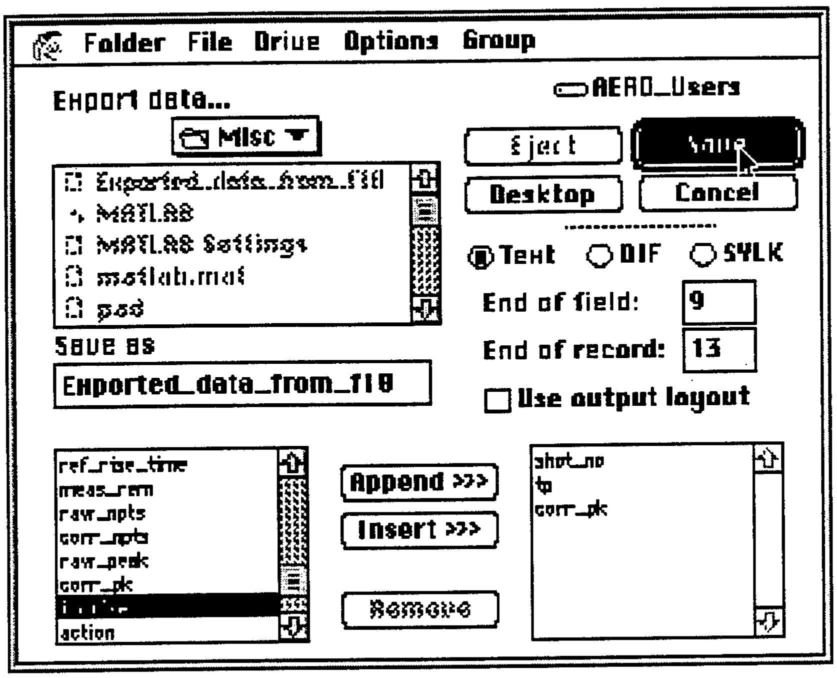


Figure 3.17: Dialog box after selecting **shot number**, **test point**, and **corrected peak**

6. Now click on the "Save" button. This will save all of the information you requested to the file name generated in the "Save as" area. This process will take a few minutes.

7. Since the data is in text format, any word processor should be able to read it. Unfortunately, the text editor **TeachText** has a 32,767 byte size restriction. The application **Microsoft Word**, however, will read the data quite readily. The application **Microsoft Word** should be located in the **Word 5.0** directory in the **Word Processing** directory of the hard disk labelled **AERO_Applications**. Launch the program now.

8. Select **Open** from the **File** menu. Find the data file you just created via the "Open" dialog box of **Word**. Open the file. The data file you opened should look like Figure 3.18. Notice that each line of text contains a shot number, followed by a "tab", followed

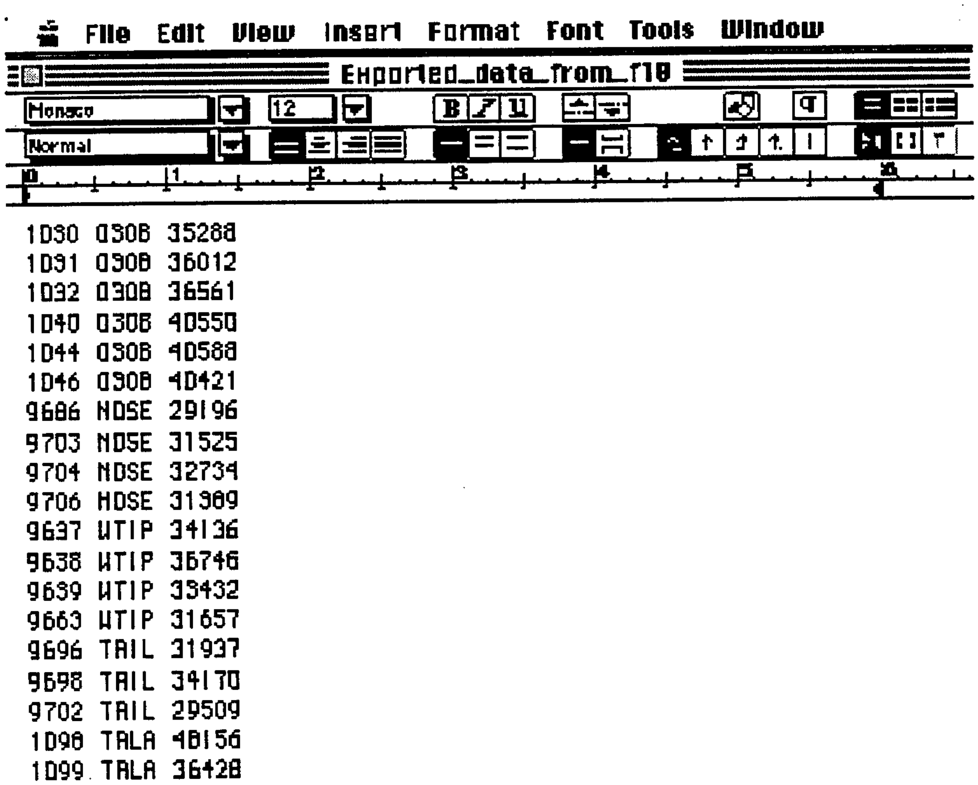


Figure 3.18: Exported data file viewed in Microsoft Word

by a test point, followed by a "tab", followed by a number (the corrected peak!). That's all there is to it. Now the data is ready to be imported to your favorite spreadsheet or word processor.

9. Quit the Microsoft Word applications.

3.4 Layout Editor

Layouts provide a "view" into the database. You use layouts to enter, modify, view and print information. This section describes the basics of creating a layout, by going through the process using an example. These are the steps required to create a layout.

1. Verify that you are in the Design environment by selecting **Design** from the **Use** menu.

2. To modify or create a layout, select **Layout** from the **Design** menu. A dialog box as shown in Figure 3.19 will appear.

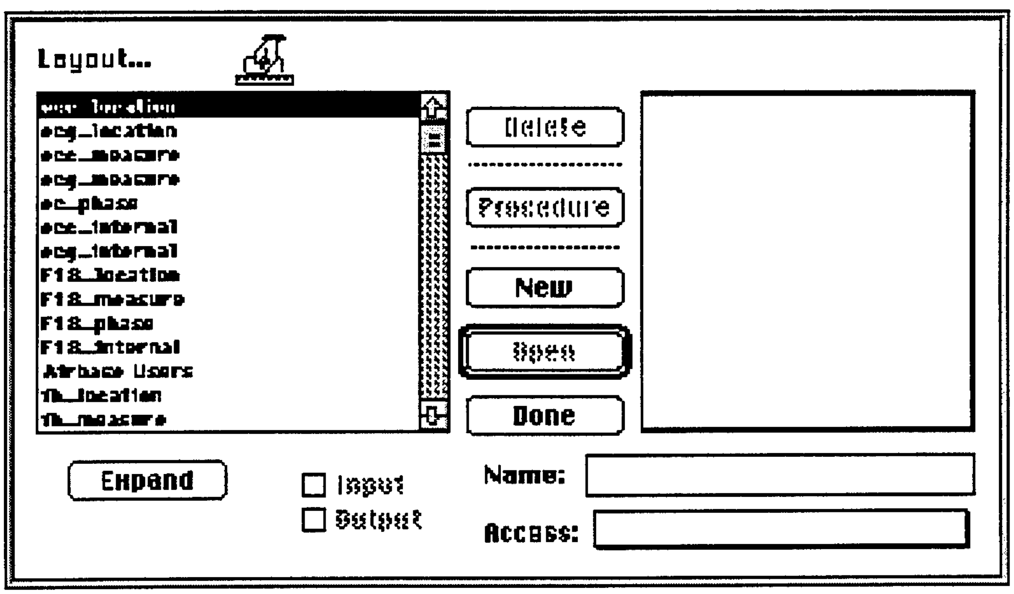


Figure 3.19: Layout Dialog Box

3. Now choose a file (select the file by clicking once on the name of the file) from the list of files located on the left side of the dialog box. For this example, choose **ecc_measure**. Next click on the **New** button in the center of the dialog box.

4. Another window, similar to the one shown in Figure 3.20 will be displayed. This dialog box allows you to create a new layout, and provides a list of the fields and files in the database. Any field from any file can be displayed in a layout. Different ways of viewing the data can be selected from the right side of the display, and the font size can also be changed. The top row of layouts is generally considered appropriate for output (display purposes, as opposed to input, where you enter new data).

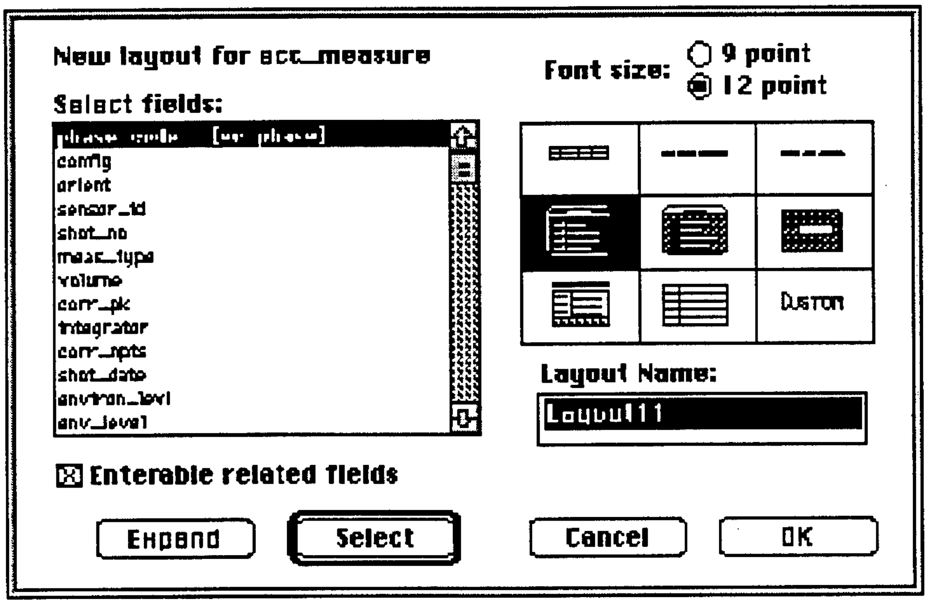


Figure 3.20: Selection of fields to include in ecc_measure dialog

5. Next, select a few fields from the left column *by selecting the item, and then clicking on the* **Select** *button.* Select **phase_code**, **shot_no**, **tp**, and **corr_pk** in that order. You may have to scroll down the list to find all of the fields. Notice that after you select each one, a number representing the order you selected the fields is displayed to the right of the selected field. If you make a mistake and select the wrong field, deselect it by simply clicking once on the field and pressing the **Select** button again. This button acts as a toggle for field selection. Note that if you *do not* select any fields, the layout will contain *all* the fields.

6. Now change the format to an output layout by choosing the top left format from the nine available choices. Change the name of the layout if you'd like, and finally click on

the **OK** button.

7. A layout like the one shown in Figure 3.21 will now be displayed. Since this layout is good enough for display purposes, close this window and switch to the **User** environment.

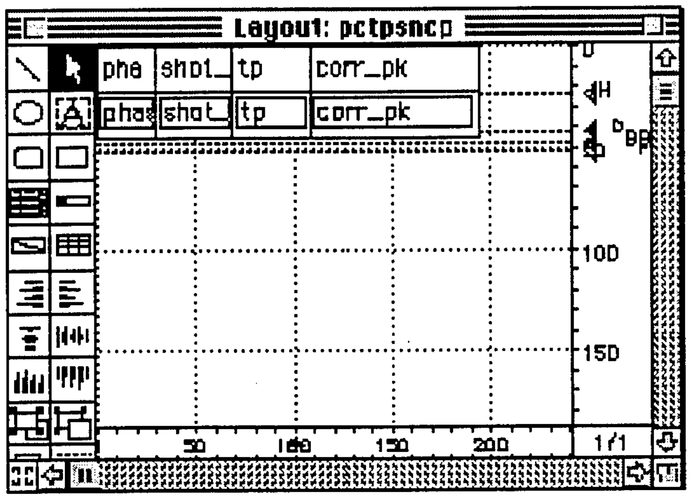


Figure 3.21: Newly created layout for ecc_measure

8. Switch to the appropriate file and layout by pressing the **command-space** key sequence, then selecting your layout from the various **ecc_measure** choices.

Chapter 4

Introduction to MATLAB

MATLAB is an acronym for MATrix LABoratory. MATLAB is an interactive software package with applications in the scientific community. The package integrates numerical analysis, matrix computations, signal processing, and graphics in an easy-to-use environment. Other benefits of using MATLAB include the inter-platform similarity. The commands you will learn for MACII–MATLAB version are virtually identical to those used on the many other operating where MATLAB can be found. These include but are *not* limited to: VAX/VMS, MS-DOS, MS-Windows, UNIX platforms (Sun, HP, VAX/Ultrix, etc.) and CRAY. Another benefit is that data files and script files created by the user on any of these platforms can be easily ported to the other systems!

The Macintosh version of MATLAB takes advantage of the inherent capabilities of the operating system. MATLAB files may have one of several icons to represent them. Five of the more important icons are shown in Figure 4.1.

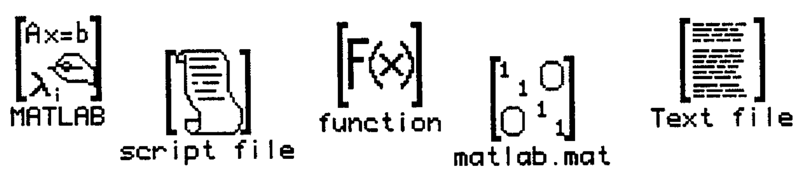


Figure 4.1: Five common MATLAB icons: Application, Script file, Function, MAT-file, Text file

4.1 MATLAB Commands

Two main windows always appear after the launch of MATLAB. The first is the Command window, where the user can type commands and view numerical results. The second is the Graphics window, where the user displays graphical output resulting from the manipulation and storage of data. MACII–MATLAB has Cut, Copy, and Paste, that work much like normal Macintosh applications.

4.1.1 Terminology

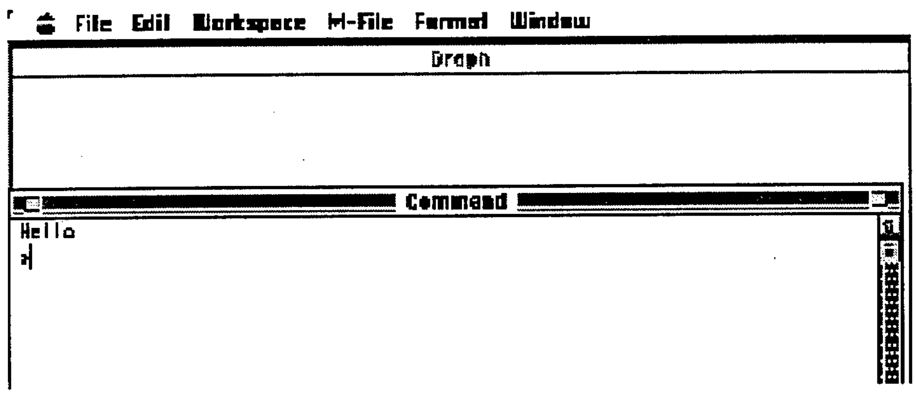


Figure 4.2: MATLAB command window immediately after launch

To begin, let us first discuss the Command window. At first it will look similar to that shown in Figure 4.2. A prompt >> is the line where information is typed. Try typing

```
>> x = 3
```

now. MATLAB will print the result

```
x =
     3
```

which says that you have assigned the number **3** to the variable x. If you don't assign a variable to the output, MATLAB automatically puts the answer in a temporary variable called ans. For example, type

```
>>3
```

and MATLAB will return

```
ans =
     3
```

thus assigning the value 3 to the temporary variable ans.

Certain variables are defined each time you start up the application. The variable pi is assigned the value 3.14159.... The variables i and j are assigned the value $\sqrt{-1}$. These variables *can* be reassigned during execution to anything you choose. Other variables which are defined are inf and NaN. These values are $\frac{1}{0}$ and $\frac{0}{0}$, respectively.

We can also assign a set of values (a vector) or a 2-dimensional set of values (a matrix) to a variable. These are examples of how to perform each.

```
>> x = [1 2 3]
x =
     1     2     3
>> z = [1 2 3; 4 5 6; 7 8 9]
z =
     1     2     3
     4     5     6
     7     8     9
```

The **space** key when used in creating a vector or matrix means *shift right one column*, and the ; means *shift down one column*. Notice the use of [and] to define the matrix. This is *usually* necessary when defining a matrix containing more than 1 element. We will discuss when it is not necessary shortly.

As these examples have shown, whenever you type a command in MATLAB, the results are displayed on the screen. Sometimes you will not want to see the results displayed on the screen (for example, if you create a matrix with values from 1 to 5000, you probably do not want to wait until MATLAB prints all the values out.) Screen output is suppressed via the ; command terminating a statement. Do not confuse this with the previously discussed use of the semicolon-colon in matrix creation. Try typing

```
>> z = [1 2 3 4];
```

MATLAB will still assign the values to a vector z, but rather than tell you it finished, it will simply return the >> prompt. If you forget to end a statement by a ; you can interrupt the display by pressing the **control-c** key sequence any time during display. The variable will still be assigned the value, but screen display will be suppressed. Note, if you hit the **control-c** sequence *during calculation*, the procedure will terminate.

MATLAB has built-in complex arithmetic. As previously discussed, the variables `i` and `j` are initially set to the engineering unit, $\sqrt{-1}$. Try typing

```
>> i
ans =
        0 + 1.0000i
```

and

```
>> y = 2+3*i
y =
   2.0000 + 3.0000i
```

At times, you may only be interested in dealing with part of a vector or matrix. *MATLAB indexes matrices starting with* 1. For example, try

```
>> a = [10 9 8 7]
a =
    10     9     8     7
>> a(1)
ans =
    10
```

The first command generated a vector, while the second asked us to return the value in the first element of the `a` vector. For a vector, the general form to return the **n**th element of vector **x** is `x(n)`. For matrices, to return the value of the element in the **n**th row and the **m**th column of matrix **x** use **x(n,m)**. We will demonstrate here

```
>> c = [1 2 3; 4 5 6]
c =
     1     2     3
     4     5     6
>> c(2,3)
ans =
     6
```

Another useful operator is the colon. You can use it to specify a range of numbers

```
>>x = 1:10
x =
     1     2     3     4     5     6     7     8     9    10
```

optionally, you can choose to give it a step size as follows

```
>> x = 0:.1:.5
x =
  Columns 1 through 7
         0    0.1000    0.2000    0.3000    0.4000    0.5000
```

The colon command is a very powerful operator; we will discuss more of its uses later. Note that the use of [and] are unnecessary in this instance. The command automatically puts them in for you.

4.1.2 MATLAB Help Facility

The help facility in MATLAB is quite good. It knows about every command available in the package. If you know the name of the command, simply type `>> help commandname`, and all of the online help for that command will be displayed on the screen. Sometimes, however, the user will not know what command he is looking for. Instead, he may have to look in the help index. The help index can be invoked by choosing **About MATLAB...** from the **Apple** menu. You will see a window like the one shown in Figure 4.3 and can navigate through the various MATLAB toolbox folders, until you find what you are looking for. MATLAB also allows you to put help in your own functions, which you will be creating later in this chapter.

4.1.3 Basic Functions

Some of the basic functions are described here. For more information on any of these functions, simply type `>> help functionname`.

- `ones(n,m)` – this function will create an **n** row by **m** column matrix of ones. If only one argument, `ones(n)` is given, the result is a square matrix of **n** rows by **n** columns.
- `zeros(n,m)` – same as `ones`, but fills matrix with zeroes.

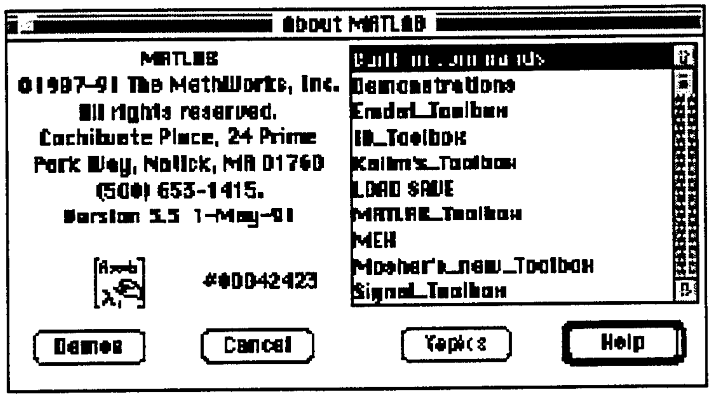


Figure 4.3: MATLAB Help Window

- `max(x)` and `min(x)` – if x is a row or column vector, the result is the maximum/minimum value of the vector. If x is a matrix, the output is a row vector with maxima/minima of each column of data.

- `sum(x)` – if x is a row or column vector, the result is the sum of all elements in the vector. If x is a matrix, the output is a row vector with sums of each column of data.

- `prod(x)` – if x is a row or column vector, the result is the product of all elements in the vector. If x is a matrix, the output is a row vector with products of each column of data.

- `size(x)` – returns the dimensions of x in a two element row vector. For example, if x contains 3 rows and 2 columns of data, the result of

```
>> size(x)
ans =
     3     2
```

- `length(x)` – returns a scalar equal to the command `max(size(x))`

- `abs(x)` – returns the absolute value of x. If x is a vector or a matrix, it returns the absolute value of every element in x.

- `phase(x)` – returns, in radians, the angle of the complex number x. If x is real, `phase(x)` returns 0. If x is a matrix, it returns a matrix of phases.

- `x'` – returns the transpose of x. If x is complex, returns the hermitian (complex conjugate transpose) of x.

- `inv(x)` – returns the inverse of x such that `x * inv(x)` is equal to the identity matrix with the same dimensions as x. Only works if x is a square matrix. In other cases, use `pinv(x)`, the pseudo-inverse.

- `ceil(x)` and `floor(x)` – returns the next highest/lowest integer value of x. Same for matrices.

- `who` and `whos` – displays the variables and optionally (`whos`) the sizes of each variable currently in memory.

- `exp(x)` , `log(x)`, `log10(x)`, `sin(x)`, `sqrt(x)` – in order of appearance: e^x, $log_e(x)$, $log_{10}(x)$, $sin(x)$, and $\sqrt{x}$

4.2 Arithmetic

MATLAB employs a very straight-forward set of arithmetic rules, allowing a novice to perform mathematical computations in the same fashion has he would intuitively.

4.2.1 Scalar Arithmetic

When working with *scalars*, the symbols "+","-", "*" , "/" , and "^" are the plus, minus, multiply, divide, and exponentiation operators. For example

```
>> a = 2 + 3
a =
     5
>> a = 2 * 4
a =
     8
>> a = 6 / 3
a =
     2
>> a = 2 - 4
```

```
a =
    -2
>> a = 2 ^ 3
a =
     8
```

In addition, when adding, subtracting, multiplying, or dividing a *matrix* by a scalar, these functions hold true. If `x` is a vector, than `x + 1` results in adding one to each element. These operations do not, however, hold for the ^ operator.

4.2.2 Vector Arithmetic

When working with two vectors, or in general, two matrices, the operations performed by the symbols above are those normally found in matrix manipulation. An error message will result if you try to perform vector operations on matrices with improper sizes. For example:

```
>> a = [1;2]
a =
     1
     2
>> b = [2;4]
b =
     2
     4
>> a * b
??? Error using ==> *
Inner matrix dimensions must agree.
```

However, the matrix operations will work just fine if the dimensions agree. In this example, we will transpose the `b` vector, thus allowing normal matrix operations to take place.

```
>> b = b'
b =
     2     4
>> a * b
ans =
```

```
     2     4
     4     8
>> b * a
ans =
    10
```

4.2.3 Element-wise Matrix Arithmetic

Often, you will need to operate on two vectors or matrices in an element-wise fashion. In other words, you want to perform an operation on each element of two matrices. In this case the "+" and "-" keys work as expected. Matrix "multiplication" and "division" and "power of" are a bit different. If the same symbols were used, MATLAB would not be able to distinguish between matrix operations and element-wise operations. Instead, MATLAB uses the operators "`.*`", "`./`" and "`.^`" to denote element-wise operations. An easy way to remember these is to think of them as the *dot operations* such as dot-product. An example of the use of these operators is as follows:

```
>> a = [1 2 3]
a =
     1     2     3
>> b = [ 4 5 6]
b =
     4     5     6
>> a .* b
ans =
     4    10    18
>> a ./ b
ans =
    0.2500    0.4000    0.5000
>> a .^ b
ans =
     1    32   729
```

Some more complex commands incorporating the simple concepts we have learned so far are incorporated into this sequence of commands:

```
>> n = 0:3
n =
     0     1     2     3
>> s = exp(j*(pi/2)*n)
s =
   1.0000                  0.0000 + 1.0000i  -1.0000 + 0.0000i  -0.0000 - 1.0000i
```

This operation applied many of the tools we have learned so far. The vector `n` was used in the exponent of the `exp` function, resulting in

$$e^{\frac{j\pi n}{2}}, \quad n = 0, 1, 2, 3$$

4.3 Graphics

MATLAB has built-in graphics. Data can be plotted in many ways such as linear, logarithmic, semi-logarithmic, contour, mesh, and polar axes. Also, you can plot multiple plots on one graph, as well as multiple graphs on a page. The most basic of these plots is the `plot()` command. If you learn how to use this, the rest of the variations are easily done by reading the help file on these commands.

4.3.1 Simple linear plots

If x is a vector, `plot(x)` plots the elements of x against their indices. To try this, execute the following commands:

```
>> x = 0:.1:pi;
>> y = sin(x);
```

You have just created a vector ranging from 0 to π with increment of 0.2 and put the results in variable x. The next step takes the $sine(x)$ and puts the results in y. Type `plot(y)`. The results are displayed in Figure 4.4. Note that the values in y are the y-values plotted against their indices. Also note that the x-axis starts at 1, not 0. *To return to the command window from the graphics window, simply press any key!*

To plot y versus x values, execute the command `>> plot(x,y)`. In general, if you pass the plot command two vectors, separated by a comma, the first vector is the x-axis, and the second vector is the y-axis. The results of this plot are shown in Figure 4.5.

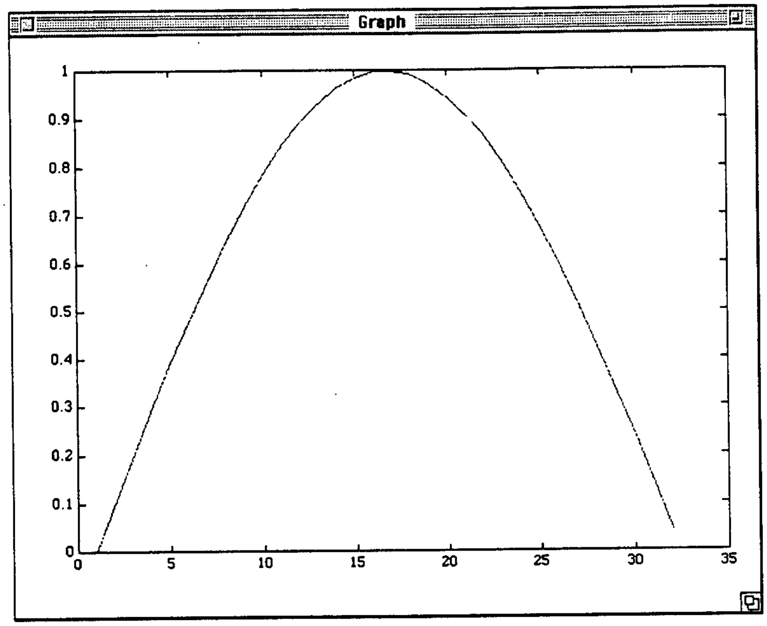


Figure 4.4: Plot of sine(x) versus its indices

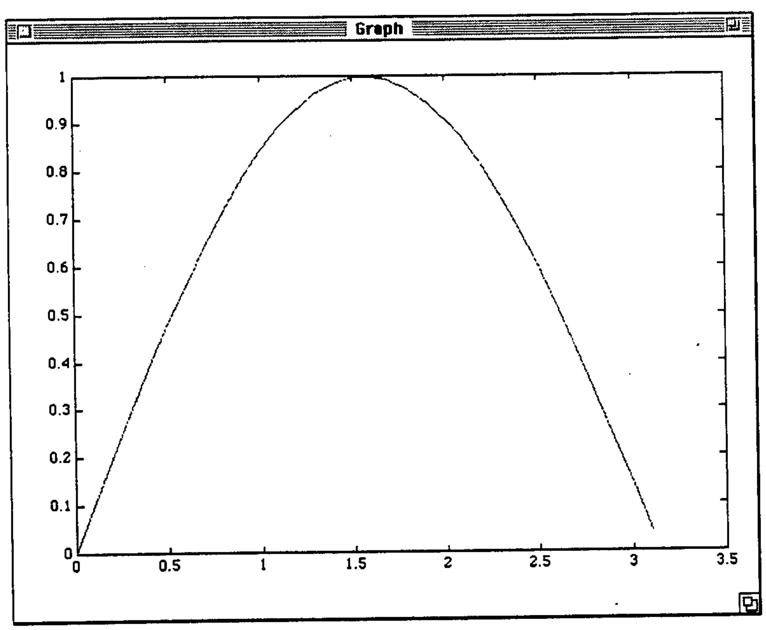


Figure 4.5: Plot of sine(x) versus x

Note: *these vectors must have the same number of points, or an error will result.*

4.3.2 Multiple Plots on the Same Axis

To plot more than one set of y values against the *same* x axis, create a matrix of the waveforms oriented in a *column-wise* fashion. For example,

```
>> x = 0:.1:2*pi;
>> y1 = sin(x);
>> y2 = cos(x);
>> y = [y1' y2'];
>> plot(x,y)
```

Notice that we transposed **y1** and **y2** when creating the **y** matrix. See Figure 4.6 for results.

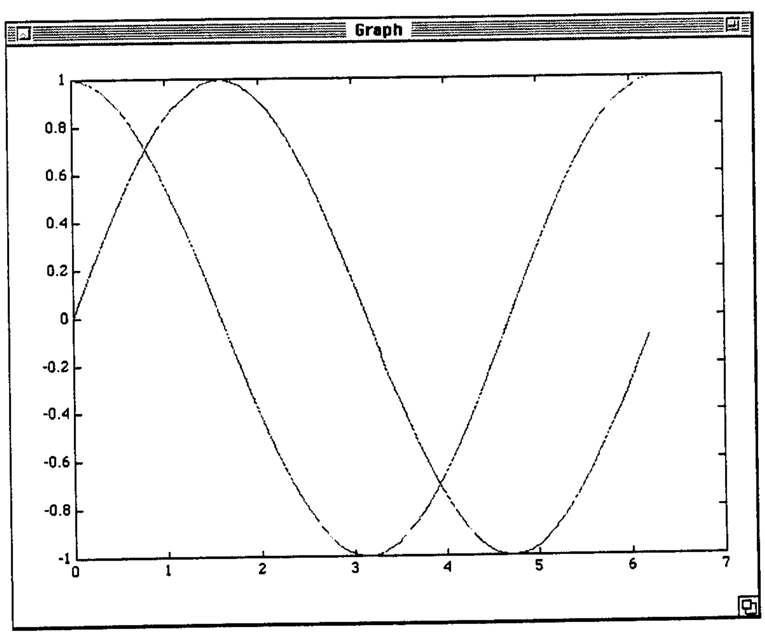


Figure 4.6: Plot of sin(x) and cos(x) on the same graph

4.3.3 Plotting Options

A third allowable parameter in the `plot` command is plotting options. The format of the command will be `plot(x, y, 'options')` where the options can be any of those discussed in Table 4.1. These commands can be mixed and matched from the columns. You may have one option from each column of the table.

Table 4.1: Plotting Options

Option	Command	Option	Command	Option	Command
solid	-	point	.	red	r
dashed	--	plus	+	green	g
dotted	:	star	*	blue	b
dashdot	-.	circle	o	white	w
		x-mark	x	invisible	i

4.3.4 Plotting with Logarithmic Axes

You can create plots with either or both axes changed to log-scale. These commands are `loglog()` (both axes logarithmic), `semilogy()` (y axis logarithmic), and `semilogx()` (x axis logarithmic). These commands use the same options as the `plot()` command.

4.3.5 Labelling Graphs

Three commands exist for this purpose. They are the `title('text')`, `xlabel('text')`, and `ylabel('text')` commands. The `text` can be any text you like. For example, try typing `>> title(' This is a test')`.

4.3.6 Multiple Graphs on a Page

The `subplot(xyn)` command will put graphics only in certain portion of the graphics window. The fields `xyn` correspond to `x` is the number of vertical divisions, `y` is the number of horizontal divisions, and `n` must be less than or equal to `x` times `y`. For example, the command `>> subplot(121)` will create two full-height, half-width areas for graphs, and selected the first (left) area as active for the first graph. After that, unless you specifically indicate which area is active, MATLAB will cycle through them with each successive plot. Typing `>> subplot` without any arguments returns MATLAB to its original, single-area state.

4.3.7 Miscellaneous Graphics Commands

Some other useful miscellaneous graphics commands are listed in Table 4.2. For more information on these commands, type `>> help commandname` in the command window.

Table 4.2: Some Graphics Commands

Command	Description
`axis`	either determines or returns axis values
`clg`	clears graphics window
`contour`	contour plot
`grid`	puts grid on graph
`hold`	holds the graph on screen (toggles on/off)
`mesh`	3 dimensional mesh-surface plot
`polar`	plots using polar coordinates
`text`	places text in graphics window

4.4 MATLAB Interface

MATLAB starts up in case-sensitive mode. If you type the command **casesen** MATLAB toggles to a case-insensitive mode.

You can clear the command window with the command `>> clc` (although MATLAB has actually just scrolled the screen up far enough to hide the previous work). No information is lost with this command. The command `>> clear` actually clears all of the variables in the workspace. The command `>> clg` clears the graphics window.

The **who** command lets the user see all of the variables defined in the workspace, and the **whos** command displays the variables and, in addition, the size of each variable.

The `>> save filename` command will save all of the variables currently in memory to the file specified. To only save a selected subset of these variables, use the modified form:

```
>> save filename var1 var2 . . .
```

The `quit` command performs the same operation as choosing **Quit** from the **File** menu–the program terminates.

Data sets, such as those saved via the `>> save filename` command can be loaded in one of two ways. If you are in the same directory as the data set, typing `>> load filename` loads the data set into memory. From anywhere, you can choose **Load...** from the **Workspace** menu. This will bring up the standard Macintosh Open File dialog box, and you can find your data set. *The data created from decompressing AIRBASE waveforms via MATLAB format is in the appropriate format to be directly read this way.* For more information on waveform decompression, see Section 3.3.5.

4.5 Signal Processing Commands

In this section we will discuss some of the more popular signal processing commands.

4.5.1 Convolution

The first of these commands is convolution. This command is executed as follows:

```
>> z = conv(x,y)
```

This command performs the discrete convolution of vector **x** with vector **y**. The analytical formula is

$$z(n+1) = \sum_{k=0}^{N-1} x(k+1)y(n-k)$$

The length of the output vector **z** will be the length of **x** plus the length of **y** minus 1.

4.5.2 Fourier Transform

The command `>> X = fft(x)` performs either the two-sided Discrete Fourier Transform (DFT) or Fast Fourier Transform(FFT) (optimized for certain data lengths), depending on the length of the vector **x**. The command `>> x = ifft(X)` performs the corresponding inverse process. The equivalent discrete operations are as follows:

$$X(k+1) = \sum_{n=0}^{N-1} x(n+1)e^{-\dfrac{j2\pi kn}{N}}$$

and the corresponding Inverse Discrete Fourier Transform is

$$x(n+1) = \frac{1}{N} \sum_{k=0}^{N-1} X(k+1) e^{\frac{j2\pi kn}{N}}$$

where N is the length of the vector x. Note that the series is written in an unorthodox fashion, running over $k+1$ and $n+1$. This is because MATLAB runs vectors from 1 to N, instead of 0 to $N-1$.

The `fft` command utilizes the FFT if the length of x is a radix-2 number, and the DFT otherwise. In other words, if the length of x is some integer power of 2 (e.g. 2, 4, 8, ..., 512, 1024, 2048, ...) MATLAB performs an FFT. The DFT is much slower for numbers greater than 32, so a radix-2 number of points is desirable.

You can pass the `fft` command an optional second parameter, `fft(x,n)`, that tells the command to zero-pad the data sequence `x` to the length `n` if `x` has fewer than `n` points. If `x` has more than `n` points, the command will only perform the FFT of the first `n` points.

The output of an FFT will, in general, be a complex sequence of points with the same length as `x`. We will use the FFT later in this chapter, during the discussion of waveform data from AIRBASE.

4.5.3 Filtering Data

Another MATLAB built-in function is called `filter`. This function is implemented as:

```
>> y = filter(b, a, x)
```

The function filters data vector `x` through a filter defined by `b` and `a` digitally. The time-domain representation of this filter is as follows:

$$\begin{aligned} y(n) &= b(1)x(n) + b(2)x(n-1) + \cdots + b(nb+1)x(n-nb) \\ &\quad - a(2)y(n-1) - \cdots - a(na+1)y(n-na) \end{aligned}$$

The corresponding frequency domain (Z-transform) description is:

$$Y(z) = \frac{b(1) + b(2)z^{-1} + \cdots + b(nb_1)z^{-nb}}{1 + a(2)z^{-1} + \cdots + a(na+1)z^{-na}} X(z)$$

4.6 Polynomial Operations

Vectors can also represent polynomials. If you want to represent an Nth-order polynomial, use a length N + 1 vector where the elements are the coefficients of the polynomial sorted in descending order of exponent. To define a polynomial with a description like

$$y = 2x^3 + 3x^2 + 1x + 4$$

you would enter `>> y = [2 3 1 4]`.

With a polynomial defined, two built-in MATLAB functions, `roots` and `poly` are useful. These functions do what you would expect (and more, but see reference [7, Chapter 3, p 164] for more details). The command `r = roots(y)` returns the roots of the characteristic polynomial `y`. The command `p = poly(r)` returns the polynomial `p` described by roots `r`. The polynomial `p` may be off from the expected result by a scaling factor, if the coefficient of the highest order factor is not 1.

4.7 AIRBASE Waveforms

4.8 Creating MATLAB Functions and Scripts

The apex of this introduction to MATLAB is to write functions and script files. Up to now, we have been executing MATLAB in a command driven mode. When a command is typed on the command (`>>`) line, it is executed immediately. MATLAB can also execute a series of commands that are stored in files. Disk files that contain one of these command sets are called *M-files.* On other platforms, MATLAB uses a ".m" extension on filenames to distinguish these files from other text files. On the Macintosh, this extension is not necessary.

The differences between a script file and a function are subtle, to be sure. A *script file* is an M-file used to automate a long sequence of commands that would normally be performed sequentially on the command line. The benefit of putting the commands in a file are that you can easily modify the entire script without having to type and execute each command over again separately. Some people choose to think of a script file as a "macro". Also, the variables used inside a script file are the same ones as in the MATLAB workspace, and can be modified in the script.

A function, on the other hand, is like the functions you have been running in MATLAB (e.g. `ones, fft, filter, poly, etc`). You may pass an input parameter or set of parameters to a function and the function may return some output parameter(s). In any case, while in the function, MATLAB does *not* have access to change or use any of the variables in your workspace.

We will create new functions and script files in the following sections.

4.8.1 Functions

In this section, you will learn how to create, modify, and save MATLAB functions. Three examples are presented here to demonstrate that functions can return no values, one value, or more than one value.

The MATLAB editor is very rudimentary (much like TeachText!). It does not allow you to change fonts, search for string occurrences, or include special characters. MATLAB programs are generally very terse, which keeps the M-files very small. The editor is more than adequate for this purpose. You can always choose, however, to write your code using another word processor, save the code in plain "text" format, then use MATLABs **Convert Text to M-file** command in the **File** menu.

Follow these steps to create a MATLAB function returning no values. This function will allow you to pass it a single number. From this number, it will generate a random sequence the length of which is determined by the number you passed. Finally, it will plot the random sequence, and give it a meaningless title!

1. Select **New** from the **File** menu. A new "Untitled" window will appear. This is the MATLAB edit window.

2. In the "Untitled" window, type the text as below. Note that all text after the `%` is only there for commenting purposes, and not necessary for the operation of the function.

   ```
   function randplot(t)
   % plots t random numbers
   x = rand(t,1);  % makes a t-row by 1-column random vector
   plot(x)
   title('meaningless title')
   ```

3. Choose **Save As...** from the **File** menu. A dialog asking you where to save the file will appear. Notice the "Save As" area has already defaulted to your filename (`randplot`).

Maneuver through the hard disk to your own work area, and then press the **Save** button.

4. Now, position the mouse pointer in the command window and click once to make the command window active. Hint: the cursor should be blinking in the command window!

5. Type the command `>> randplot(20)`. A screen similar to Figure 4.7 should appear.

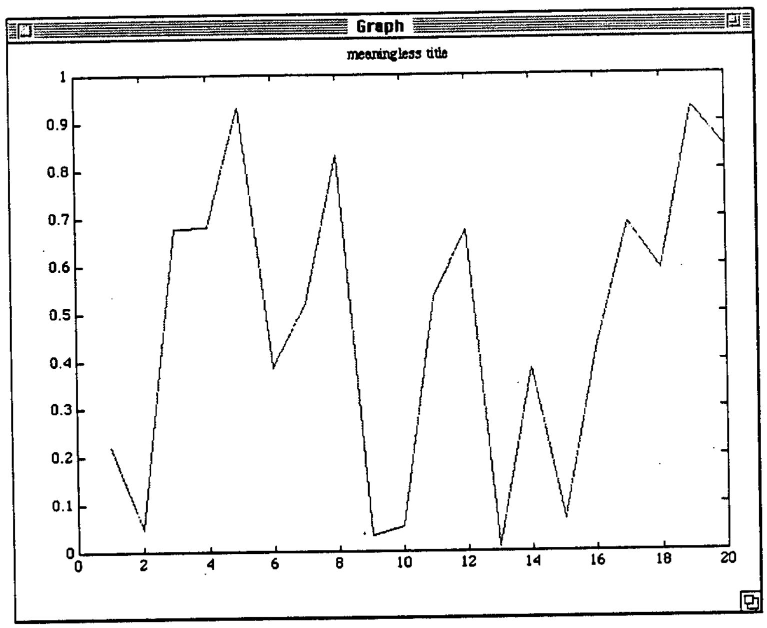


Figure 4.7: Result of **randplot(20)**

6. On a side note, any commented lines in the function (lines beginning with a `%`) that come before the first line of actual code are used for the `help` command. Try typing `>> help randplot` now.

A second function, `fact`, will return the factorial of any number you pass it. For example, if you type `fact(5)` the command will calculate $5 * 4 * 3 * 2 * 1$. This is almost a trivial problem, and can be solved a number of ways. Here is one way:

1. Create a new M-file as in the previous example.

2. Type the text exactly as below (commented lines not necessary, of course)

```
function y = fact(n)
% this function computes the factorial of n
y = 1;
for i = 2:n
 y = y*i;
end
```

3. Save the file in the same fashion as you did earlier.

4. Try it with a few values!

```
>> a = fact(4)
a =
    24
>> b = fact(10)
b =
      3628800
>> c = fact(20)
c =
   2.4329e+18
>> fact(12)
ans =
   479001600
```

5. Note that if you don't assign the returned value, it puts it in the temporary variable ans just as you would expect.

The next function will has two return values. You pass it a parameter, just as the last two examples. Again, this is almost a trivial example, but it demonstrates at least two important concepts.

1. As in the previous examples, create a new "Untitled" window.

2. Type the routine as below:

```
function [x,y] = factgauss(n)
```

```
% This function returns factorial of n in x
% Also returns the sum of 1 to n using
% gauss' method of determination
x = fact(n);     % note, calling another function
y = n*(n+1)/2;  % gauss figured this out in grade school!
```

3. First of all, notice that we computed x by calling the old program `fact`. It is natural in MATLAB to call one function from another function.[1]

4. Now, let's run this function in several ways ..., first

```
>> factgauss(5)
ans =
    120
```

5. next try ...

```
>> a = factgauss(5)
a =
    120
```

6. and finally ...

```
>> [a,b] = factgauss(5)
a =
    120
b =
     15
```

7. Notice that in the first two cases, only the factorial was returned. You must explicitly request multiple return values, or only the first value will be returned. Also, notice in the last one that when we request multiple return values, we must surround the variables by `[ ]`, and separate them by a comma.

[1] For any programmer-types who may be reading this document, a note on recursion. Recursion is allowed, *but not recommended.* Deeply nested recursive loops cause MATLAB to crash!

4.8.2 Script Files

Lastly, we will create a simple script file. This script generates the fibonacci numbers between 1 and 1000, and then plots them on the screen.

A bit of historical trivia on the Fibonacci numbers. Liber abaci, Leonardo of Pisa, known also as Fibonacci posed the *rabbit problem.* Rabbits obey the following law of breeding: "Every pair of rabbits (at least two months old) will produce one pair of rabbits as offspring every month." Fibonacci formulated the series:

$$F_n = F_{n-1} + F_{n-2}$$

where $F_1 = F_2 = 1$. This problem is solved quite easily with the following script file.

1. In the command window, type >> `clear` to clear the contents of MATLAB's workspace. Type >> `whos` to verify no variables have been declared.

2. Create a new "Untitled" window.

3. Type the text as below:

```
% This M-file calculates Fibonacci numbers
f = [1 1]; i = 1;
while f(i) + f(i+1) < 1000
f(i+2) = f(i) + f(i+1);
i = i + 1;
end
plot(f)
title('fibonacci series')
```

4. Save this file as `fibonacci`. Note the use of the `while` loop statement. MATLAB also has built-in `for` loops and `if` statements. For more information on these control flow statements, see reference [7]pp. 2-77 to 2-80

5. Run the program by typing >> `fibonacci` from the command window. This results in a plot similar to that shown in Figure 4.8. Notice that you passed no parameters and the script file returned no parameters. However, typing >> `whos` now will show that two variables have been created in the workspace. *These variables are the same*

as those created in the script file. The most important difference between functions and script files is the ability of script files to take advantage of MATLAB's ability to make use of workspace variables.

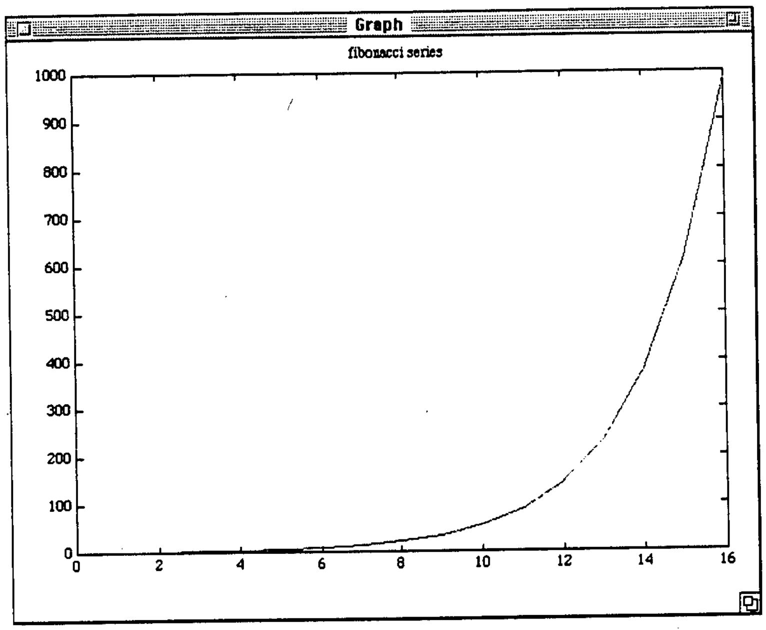


Figure 4.8: Output of script file "fibonacci"

4.9 MATLAB Search Paths

You may be asking yourself "where does MATLAB look when I type a function name on the command line?" The default search order where MATLAB will search for a function or command is:

- Is the command a variable in memory?
- Is the command a built-in function?
- Is the command a function in the present working folder?
- Is the command a function in the MATLAB search path?

The MATLAB search path is modifiable, but it originally defaults to all the folders in the MATLAB directory. To find out what folders are set on your machine, select **Set Path**...from the **M-file** menu. A list of pathnames like the one shown in Figure 4.9 should appear. These paths may be modified to include alternate paths. For example, if you choose

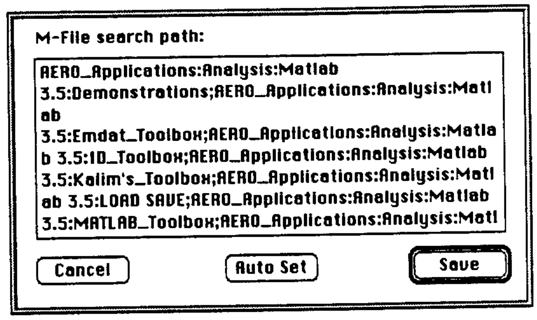


Figure 4.9: MATLAB Search Paths dialog box

to include a working directory called **My_Toolbox** in your user directory (for example **User_Kalim** on the **AERO_Users** hard disk), add the path

`;AERO_Users:User_Kalim:My_Toolbox`

The ; separates your path from the other paths.

4.10 AIRBASE Data

In Section 3.3.5 we discussed decompression of AIRBASE waveforms. If you did not perform decompression at that time, refer to that section and decompress a waveform now. If MATLAB is still running, quit the program, either by typing `>> quit` from the command line or by selecting **Quit** from the **File** menu.

After you decompressed a waveform, you should see its icon somewhere on the desktop (exactly where depends on where you chose to decompress), the default being in the AIRBASE directory. Follow these procedures to learn a bit more about the AIRBASE waveforms. We will be performing some of the operations learned earlier in this chapter on real data.

1. Double-click on a decompressed file (I chose 0560AZP6.HP for this example.) Notice that even though the file was created with 4th DIMENSION, the "double-click" operation has the effect of launching MATLAB. This is a feature of the decompression

routine. The data is now loaded into the MATLAB workspace just as though it was previously saved by MATLAB as a MAT-file.

2. Type `>> whos` on the command line. The variables corresponding to Table 4.3 should be displayed along with their corresponding sizes. The most important variables for

Table 4.3: Standard Data Format after Decompression to MATLAB format

Variable Name	Description
`ar`	Autoregressive coefficients
`bindata`	Encoded residuals from compression algorithm
`header`	23-byte header
`timdata`	Time axis
`variance`	Variance (scaling for reconstruction)
`ydata`	Reconstructed waveform

analysis are the `ydata` and `timdata` variables (the `header` may be necessary for reference.)

3. Now plot the data using the command `>> plot(timdata,ydata)`. This plots the data versus its time axis (stored in the vector `timdata`).

4. Next you will perform a Fourier Transform of the data. You could simply type the command `>> Y = fft(ydata);` but if `ydata` has a length other than a radix-2 value (as explained in Section 4.5.2) you will want to use the modified version of the command `>> Y = fft(ydata,npts)` where `npts` is first radix-2 value greater than the number of points in the waveform. You will recall that to determine the length of a vector, e.g. *ydata*, you type the command `length(ydata)`. For example, in the file **0560AZP6.HP**, `ydata` has 796 points, so the command would be `>> Y = fft(ydata,1024)` since 1024 is the next radix-2 value.

5. In general, `Y` will be complex. Usually, we are interested only in viewing the magnitude (although the phase has its uses as well). We can plot the resulting magnitude using the command `>> semilogy(abs(Y))` which plots the magnitude of `Y` against its indices. The corresponding output of my data file (0560AZP6.HP) is shown in Figure 4.10.

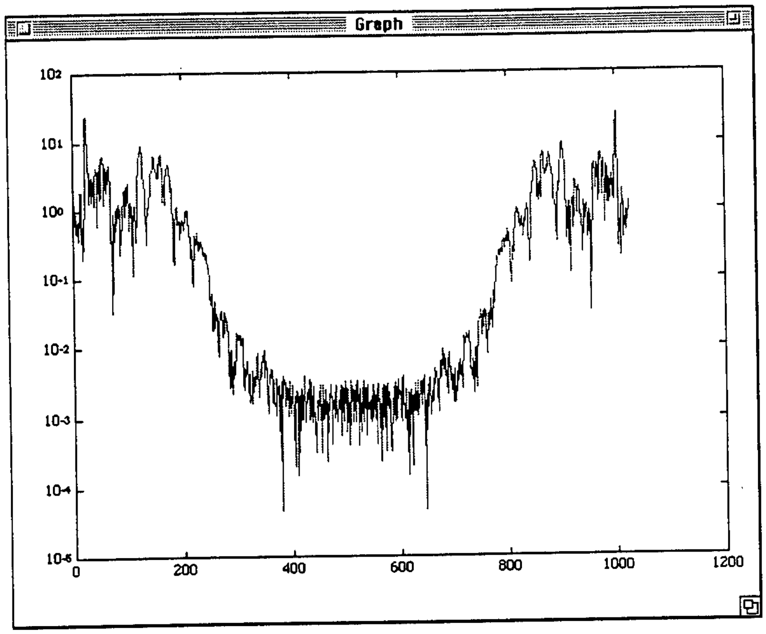


Figure 4.10: Frequency plot: 2-sided spectrum - 0560AZP6.HP

6. Notice the symmetry. The first half of the plot is the positive frequency range from DC to F_{max}, followed by the maximum negative frequency component, $-F_{max}$. The last point is the smallest negative frequency component. Since the positive and negative frequency components are conjugate symmetric by definition of the Fourier Transform of real data, we need only examine the first half of the data sequence. To plot this, type `>> semilogy(abs(Y(1:512)))`

7. The first half of the data set is plotted, with frequency ranging from DC to F_{max}. One last problem is still lingering, however. The x axis is only the indices, instead of the actual frequencies we want. In theory, if we know δt (the sample interval of an evenly spaced data set), the maximum frequency is $F_{max} = \frac{1}{2 \cdot \delta t}$

8. A frequency axis can be generated many ways, and this is one of them. First off, we must find δt. We know the entire time series, so we can just take the difference between two neighboring time points in the series as follows:

```
>> delT = timdata(2) - timdata(1)
delT =
```

```
    2.0000e-09
```

9. F_{max} is easily derived via the command:

```
>> Fmax = 1 / (2 * delT)
Fmax =
    2.5000e+08
```

10. The next step is to create a frequency axis ranging from 0 to 1 that has the appropriate number of points. We will do this so that we can next multiply this sequence by F_{max} and thus range from 0 to F_{max}. The two steps are as follows:

```
>> frequency = (0:511) ./ 512;
>> frequency = frequency * Fmax;
```

11. Finally, we will plot the data versus the proper axes, with the following command:

```
>> semilogy(frequency, abs(Y(1:512)))
```

This concludes the introduction to MATLAB. This was only meant as a brief introduction, and is in no way a comprehensive guideline of the functionality of the program. People are discovering more uses for this package daily, and literature on the subject should be commonly available. MATLAB is used widely in academia as well as in commercial industry, as many professionals are discovering the need to write their own routines in C and FORTRAN unnecessary and wasteful.

The routines written by the MATLAB designers are functions, just like the ones you have learned to write. They are readable, and a great deal can be learned by perusing some of these codes. For example, take a look at the `Hilbert` transform routine some time ...

Chapter 5

AIRBASE Data Format

This chapter describes the format used in the AIRBASE database design. In all cases, the database contains the phase, location, internal and measurement files. The "core" set of files are the four mentioned above.

The following sections describe the contents of each file type, and the associated field types and sizes. These summary tables (Tables 5.1 through 5.6) are taken from the AIRBASE Final Report [1, Appendix A].

The format is to give the file name, a description, then a list of the columns and their type and attributes, and finally a description of each individual column. Included under the "Type and Attribute" heading are key interrelations and indexing information. "Column Descriptions and Codes" includes a list of any codes and their meanings.

5.1 Test Point Location Table.

The location table gives physical locations of each test point of an assessment, sorted by test point.

Column Descriptions and Codes:

1. **Test_point_id**
    - A four character alpha-numeric index associated with each physical location of a test point and measurement volume.
    - Code: none.
2. *volume_id*

- A two character code giving the test volume. This field differentiates between internal (wire and bulk), external (skin), and free field. Beyond this, new codes may be added for POE and varying levels of shield enclosure.

Code_string	Description
IN	Internal measurement
PO	Internal POE
EX	External skin
FF	Free Field

3. *Subvolume_id*

   - For internal, external, and other aircraft associated test points, this six character mnemonic gives a rough location of a test point per the aircraft drawings. It can be used to compute distances in a rough manner also, but is most useful for binning results by aircraft section. For free field test points, this will distinguish between the various facilities.

   - Some codes for this field will be assessment specific since each aircraft is structured differently. The codes for the free field test points can be standardized.

Code_string	Description
HPD	HPD
VPD	VPD
TRE	TRESTLE
CWV	CW
LNOSE	Left hand nose
RNOSE	Right hand nose
L2BAY	Left hand #2 bay
R2BAY	Right hand #2 bay
L3BAY	Left hand #3 bay
R3BAY	Right hand #3 bay
NOSWW	Nose wheel well
LFCOCK	Left forward cockpit
RFCOCK	Right forward cockpit
LACOCK	Left hand aft cockpit
RACOCK	Right hand aft cockpit
LLEEXT	Left hand leading edge extension

RLEEXT	Right hand leading edge extension
LCNTFU	Left hand center fuselage
RCNTFU	Right hand center fuselage
LAFTFU	Left hand aft fuselage and fin
RAFTFU	Right hand aft fuselage and fin
LYING	Left hand wing
RWING	Right hand wing
WPYLON	Wing pylon
CPYLON	Centerline pylon
MISC	Miscellaneous multiple use items

4. *X*
   - For internal, external, and other aircraft associated test points this corresponds to station. For free field test points this is x on the facility grid.
   - Code: none.
5. *Y*
   - For internal, external, and other aircraft associated test points this corresponds to waterline. For free field test points this is y on the facility grid.
   - Code None.
6. *Z*
   - For internal, external, and other aircraft associated test points this corresponds to buttockline. For free field test points this is z on the facility grid.
   - Code None.
7. *Location_remarks*
   - Remarks concerning the location.
   - Code None.

Table 5.1: Table Location: Columns and Attributes.

Column Name	*Type*	*Attributes*
test_point_id	CHAR(4)	NOT NULL, primary key
volume_id	CHAR(2)	NOT NULL
subvolume_id	CHAR(6)	
x	NUMBER	
y	NUMBER	
z	NUMBER	
location_remarks	CHAR(40)	

5.2 Test Phase Table

The test phase table gives information about each test phase. The test phase is a unified segment of the test which has the same experiment, configuration, orientation, etc. That is, the test item is physically unperturbed throughout a phase, and the purpose of testing remains constant. For some assessments, it may be necessary to create new experiment numbers (e.g., for field mapping on F/A-18) which were not present in the original assessment.

Table 5.2: Test Phase Table: Columns and Attributes.

Column Name	*Type*	*Attributes*
phase_code	CHAR(3)	NOT NULL, primary key
experiment	CHAR(2)	
configuration_code	CHAR(3)	
orientation_code	CHAR(2)	
facility_code	CHAR(2)	
facility	CHAR(2)	
phase_position	CHAR(1)	
phase_power	CHAR(1)	
phase_name	CHAR(16)	
phase_description	CHAR(80)	
object_orientation	NUMBER	
object_mode	CHAR(2)	
object_position_x	NUMBER	
object_position_y	NUMBER	
object_position_z	NUMBER	
pulser_id	CHAR(6)	
stores	CHAR(2)	
phase_remarks	CHAR(40)	

Column Descriptions and Codes:

1. **Phase_code**

   - This three character field indicates a unique combination of experiment, configuration, orientation, etc. It reflects the test design. It is primary key of the phase table and a foreign key of the measurements table. The first two characters represent the (assessment specific) experiment number, which can be used to reference test documents or an AIRBASE help text file. The last character represents variations which occurred within the experiment starting with "a", and going to "z".

- Code None.

2. *Experiment*

    - The two character experiment corresponds to the experiments found in the pretest and post-test documentation.
    - Code None.

3. *Configuration_code*

    - This is the original three character configuration code used on the test and stored in the header.
    - Code Test Specific.

4. *Orientation_code*

    - This is the original two character orientation code used on the test and stored in the header.
    - Code Test Specific.

5. *Facility_code*

    - This is the original two character facility code used on the test and stored in the header. It was unique to each aircraft.
    - Code Test Specific.

6. *Facility*

    - This indicates the facility for a phase.
    -

Code_string	Description
HP	HPD
VP	VPD
TR	TRESTLE
CW	Continuous wave

7. *Phase_position*

    - Phase position refers to whether the aircraft was on or off the test stand.

Code_string	Description
0	On ground
1	On test stand

8. *Phase_power*

   - Indicates whether power on or off during the phase. Further details as to how power was applied are reserved for phase_description.
Code_string	Description
0	Power off
1	Power on

9. *Phase_name*

   - Optional field for identifying phase.
   - Code None.

10. *Phase_description*

    - Field for describing phase.
    - Code None.

11. *Object_orientation*

    - Gives the angle between the aircraft longitudinal axis, taken to be positive towards the nose, and the facility grid z axis in a counterclockwise direction. This forms a standard right handed coordinate system (rotate z into x giving y as the vertical). For HPD, the z axis points along the access road away from the pulser, with the x axis parallel to the polarization vector pointing west. For VPD, z points away from the pulser, as is the case for TRESTLE.
    - Code None.

12. *Object_mode*

    - Indicates the mission phase for the aircraft which the particular phase and experiment are trying to simulate.

Code_string	Description
GA	Ground Alert
CA	Carrier Alert
TX	Taxi
TO	Takeoff
CR	Cruise
FU	In Flight Refuel
AT	Attack
SP	Storm penetration

13. *Object_position_x*

    - Location of a precisely defined point on the test item along facility x-axis, in meters.
    - Code None.

14. *Object_position_y*

    - Location of a precisely defined point on the test item along facility y-axis, in meters.
    - Code None.

15. *Object_position_z*

    - Location of a precisely defined point on the test item along facility z-axis, in meters.
    - Code None.

16. *Pulser_id*

    - Six character acronym for the pulser used for the test.
Code_string	Description
HAGIIC	AFWL's HAG-IIC pulser

17. *Stores*

    - Indicates the number and type of stores attached for a particular phase.
    - Codes will be assessment specific in many cases. However, some of the more common ones will be standardized.

Code_string	Description
00	No stores attached.
01	Fully loaded, conventional.
02	Fully loaded, conventional and nuclear mix.
04	Outer stores only.
05	Inner stores only.
11	F/A-18A B-61 weapons configuration.

18. *Phase_remarks*

- Indicates success or failure of experiment, whether it was an add on, etc.
- Code None.

5.3 Internal Test Point Description Table

The internal test point descriptions table gives information on internal test points that are associated with aircraft hardware. Descriptions of the hardware are specific, but a few general columns like "entity_code" divvy up the wire types into three broad categories for binning studies.

Table 5.3: Internal test point description table.

Column Name	*Type*	*Attributes*
test_point_id	CHAR(4)	NOT NULL, primary key, foreign key from Table Location.
measured_entity	CHAR(2)	
test_point_configuration	CHAR(2)	
number_wires_measured	NUMBER	
wire_id	CHAR(15)	
entity_code	CHAR(1)	
subsystem	CHAR(3)	
subsystem_number	CHAR(12)	
entity_function	CHAR(3)	
entity_length	NUMBER	
number_of_branches	NUMBER	
number_of_segments	NUMBER	
lru_name	CHAR(20)	
lru_equipment_id	CHAR(10)	
lru_military_pn	CHAR(10)	
lru_manufacturer_pn	CHAR(20)	
lru_manufacturer	CHAR(20)	
lru_relationship	CHAR(2)	
lru_connector_p_id	CHAR(10)	
lru_connector_j_id	CHAR(10)	
lru_connector_part_id	CHAR(20)	
lru_pin_id	CHAR(4)	
number_of_pins	NUMBER	
lru2_equipment_id	CHAR(10)	
lru2_relationship	CHAR(2)	
lru2_connector_p_id	CHAR(10)	
lru2_connector_j_id	CHAR(10)	
lru2_pin_id	CHAR(4)	
impedance_real	NUMBER	
impedance_imag	NUMBER	
internal_remarks	CHAR(40)	

Column Descriptions and Codes:

1. **Test_point_id**

   - A four character alpha-numeric index associated with each physical location of a test point and measurement volume.
   - Code None.

2. *Measured_entity*

   - Wire, coax center conductor, twisted pair, etc.
   -

Code_string	Description
01	Single wire, unshielded
02	Single wire, shielded
03	Twisted pair, unshielded
04	Twisted pair, shielded
05	Twisted triple, unshielded
06	Twisted triple, shielded
07	Coax
08	Twinax
09	Triax
10	Multi-conductor, unshielded
11	Multi-conductor, shielded

3. *Test_point_configuration*

   - What the measured entity is a part of. Uses the same codes as measured_entity. E.g., if the test point was a single unshielded wire protruding from a twisted shielded pair, the measured_entity would be 01 while the configuration would be 04.
   -

Code_string	Description
01	Single wire, unshielded
02	Single wire, shielded
03	Twisted pair, unshielded
04	Twisted pair, shielded
05	Twisted triple, unshielded
06	Twisted triple, shielded
07	Coax

08	Twinax
09	Triax
10	Multi-conductor, unshielded
11	Multi-conductor, shielded

4. *Number_wires_measured*
   - Gives number of wires in multi-conductor group which are being measured.
   - Code None.
5. *Wire_id*
   - Wire identifier per test item wiring diagrams.
   - Code None.
6. *Entity_code*
   - Gives a general indication of the type of entity measured. Inferred from wiring diagrams using definition of a cable from MIL-5088J.
   -

Code_string	Description
M	Multiconductor group. Applies to several wires or cables not within the same jacketing within the vicinity of the test point.
C	Cable. Applies to several wire wires or groups of wires which are within the same jacket in the vicinity of the test point.
W	Wire. A conductor which carries a single signal; e.g., a twisted pair or coax, or a single wire.

7. *subsystem*
   - Indicates subsystem with which the test point is associated.
   - Many codes will be assessment specific. However, some can be standardized and are given below.

Code_string	Description
ARM	Armament
COM	Communications
UHF	UHF Radio
RDR	Radar
CON	Control surfaces
ENG	Engine control
DC	DC power
AC	AC power
FUE	Fuel
IGN	Ignition
ENV	Environmental (heating, etc.)

8. *subsystem_number*

   - Part number for subsystem, e.g. AN/ARC-164 for UHF.
   - Code None.

9. *Entity_function*

   - Gives a broad category for the type of signal which the entity carries.

Code_string	Description
PWR	Power
GND	Ground or return
MIX	Mixed
ANA	Analog signal
ANT	Antenna
DGD	Digital data signal
DGC	Digital control signal

10. *Entity_length*

   - Gives an exact or estimated length of the entity as obtained from wiring diagrams. Individual wires usually have exact lengths associated with them, while cables (bulks) usually are associated with average lengths.

- Code None.

11. *Number_of_branches*
    - Indicates number of branches from the entity.
    - Code None.

12. *Number_of_segments*
    - Indicates number of connector-to-connector segments along the wire.
    - Code None.

13. *lru_name*
    - Name of LRU (WRA) associated with test point (usually nearest one), e.g. UHF receiver/transmitter.
    - Code None.

14. *lru_equipment_id*
    - Reference designator for LRU.
    - Code None.

15. *lru_military_pn*
    - Military part number for LRU, e.g. RT-1146/ARC-164.
    - Code None.

16. *lru_manufacturer_pn*
    - The manufacturer's part number for the specific piece of equipment, e.g. 706173-802.
    - Code None.

17. *lru_manufacturer*
    - Vendor's name, e.g. Magnavox.
    - Code None.

18. *lru_relationship*
    - Describes the location of the LRU of interest with respect to where the actual measurement was made.

Code_string	Description
P	Primary. Measurement made at LRU/connector interface.
A	Associated. Measurement made at LRU2.

19. *lru_connector_p_id*

    - Plug ID from wiring diagrams.
    - Code None.

20. *lru_connector_j_id*

    - Jack ID from LRU circuit diagrams.
    - Code None.

21. *lru_connector_part_id*

    - MIL part number from manufacturer's parts lists. If non-MIL part, use part number from manufacturer's parts list.
    - Code None.

22. *lru_pin_id*

    - ID of the particular pin being measured.
    - A few codes are necessary for those entities which are not single wire measurements.

Code_string	Description
BULK	The entire connector response is to be measured.
GRUP	Only a group of pins is measured. The list of which pins are measured is given in the remarks. If the group is a particular signal type, the signal type can be garnered from entity_function.

23. *Number_of_pins*

- Number of pins in connector.
- Code None.

24. *Lru2_equipment_id*

- Equipment id for LRU2, the terminating (end of wire) or associated (alternate point of measurement) box.
- Code None.

25. *Lru2_relationship*

- Describes the location of LRU2 with respect to where the actual measurement was made.
- Code_string / Description

Code_string	Description
T	Terminating. End of wire makes contact at lru.
A	Associated. Measurement made at LRU2.

26. *Lru2_connector_p_id*

- Plug ID from wiring diagrams.
- Code None.

27. *Lru2_connector_j_id*

- Jack ID from wiring diagrams.
- Code None.

28. *Lru2_pin_id*

- ID of the particular pin being measured.
- A few codes are necessary for those entities which are not single wire measurements.

Code_string	Description
BULK	The entire connector response is to be measured.

GRUP	Only a group of pins is measured. The list of which pins are measured is given in the remarks. If the group is a particular signal type, the signal type can be garnered from entity_function.

29. *Impedance_real*
    - Estimated input impedance of interface circuit associated with the test point, real part.
    - Code None.
30. *Impedance_imag*
    - Estimated input impedance of interface circuit associated with the test point, imaginary part.
    - Code None.
31. *Internal_remarks*
    - Additional information about the internal test points.
    - Code None.

5.4 Measurement Record Description Table

This table contains information on each data record from an assessment. Note that data records which have been relegated to "BAD" quality status during the assessment are not included in this table; these will be retained only in tape archives.

Column Descriptions and Codes:

1. **test_phase_code**

   - This three character field indicates a unique combination of experiment, configuration, orientation, etc. It reflects the test design. It is primary key of the phase table and a foreign key of the measurements table. The first two characters represent the (assessment specific) experiment number, which can be used to reference test documents or an AIRBASE help text file. The last character represents variations which occurred within the experiment, and runs from "a" to "z".
   - Code None.

2. **Sensor_id**

   - A unique identifier associated with each probe for a given assessment. Used as index in Measurements Table for multiprobed test points.
   - Code None.

3. **Test_point_id**

   - A four character alpha-numeric index associated with each physical location of a test point and measurement volume.
   - Code None.

4. **Shot_number**

   - Four digit number which indicates the facility shot number.
   - Code None.

Table 5.4: Measurement Description Table: Columns and Attributes.

Column Name	*Type*	*Attributes*
test_phase_code	CHAR(3)	NOT NULL, composite primary key foreign key from table Phase.
sensor_id	CHAR(4)	composite primary key, relates to probe ids in Calibration Data table.
test_point_id	CHAR(4)	NOT NULL, composite primary key foreign key from table Location and Internal.
shot_number	CHAR(4)	NOT NULL, composite primary key.
facility	CHAR(2)	NOT NULL
measurement_type	CHAR(2)	
volume_id	CHAR(2)	
measurement_units	CHAR(10)	
integrator_id/differentiator_id	CHAR(4)	
data_link_id	CHAR(4)	
filter_id	CHAR(4)	
signal_conditioner_1	CHAR(4)	
signal_conditioner_2	CHAR(4)	
signal_conditioner_3	CHAR(4)	
signal_conditioner_4	CHAR(4)	
total_raw_data_points	NUMBER	
total_corrected_data_points	NUMBER	
corrected_sample_rate	NUMBER	
acquisition_instrumentation	CHAR(2)	
shot_time	CHAR(6)	
shot_date	CHAR(6)	
environmental_level	NUMBER	
reference_rise_time	NUMBER	
raw_peak_amplitude	NUMBER	
corrected_peak_amplitude	NUMBER	
corrected_impulse	NUMBER	
corrected_action	NUMBER	
corrected_rectified	NUMBER	
corrected_impulse_peak	NUMBER	
corrected_max_rise	NUMBER	
corrected_measurement_record	TEXT	
measurement_remarks	CHAR(40)	

5. *Facility*

- This indicates the facility for a test phase.
Code_string	Description
HP	HPD
VP	VPD
TR	TRESTLE
CW	Continuous wave

6. *Measurement_type*

- Two character code to indicate the type of measurement made. For external and free field volumes, this differentiates between the H and E fields etc. For internals, the measurement type distinguishes between voltage or current measurements on a particular test point, and also discriminates among gross categories of measurement entities, such as bulk, wire signal or ground. More specific information along these lines can be found in Table Internal.
Code_string	Description
SH	Shield current
IS	Short circuit current
IW	Individual wire current
IB	Bulk current
IG	Ground current
IC	Coax center conductor current
BO	Bulk over shield
CO	Coax over shield
VO	Open circuit voltage
VL	Loaded voltage
ZS	Impedance, source
ZL	Impedance, load
B1	X-polarized magnetic field
B2	Y-polarized magnetic field
B3	Z-polarized magnetic field
B4	X-polarized derivative magnetic field
B5	Y-polarized derivative

	magnetic field
B6	Z-polarized derivative magnetic field
B7	XZ-polarized magnetic field
B8	XZ-polarized derivative magnetic field
E1	X-polarized electric field
E2	Y-polarized electric field
E3	Z-polarized electric field
E4	X-polarized derivative electric field
E5	Y-polarized derivative electric field
E6	Z-polarized derivative electric field
E7	XZ-polarized electric field
E8	XZ-polarized derivative electric field
J1	Axial surface current density
J2	Circumferential surface current density
J3	Derivative axial surface current density
J4	Derivative circumferential surface current density
Q1	Surface charge density
Q2	Derivative surface charge density
IT	RESERVED
ID	RESERVED
NA	Ambient noise
NC	Coherent noise (due to pulser)
NS	EGG ODS noise check

7. *Volume_id*

- A two character code giving the test volume. This field differentiates between internal, (wire and bulk), external (skin), and free field. Beyond this, new codes may be added for POE and varying levels of shield enclosure.

Code_string	Description
IN	Internal measurement
PO	Internal POE
EX	External skin
FF	Free Field

8. *Measurement_units*

   - Units represented by the measurement record.
Code_string	Description
VOLT	Volts
AMPS	Amperes
C/M2	Coulombs per square meter
A/M	Amperes per meter
KV/M	KiloVolts per meter
TESL	Tesla
OHMS	Ohms
F	Farads
H	Henries
DEGR	Degrees
RADN	Radians
KV/M/NS	Kilovolts per meter per nanosecond
RADS	Rads silicon

9. *Integrator_id/Differentiator_id*

   - A unique identifier associated with each analog integrator, usually used for free field or external measurements.
   - Code None.

10. *Data_link_id*

    - A unique identifier associated with each data link.

- Code None.

11. *Filter_id*

    - A unique identifier associated with each analog filter used to roll off data for digitization.
    - Code None.

12. *Signal_conditioner_1*

    - A unique identifier for an additional signal conditioner.
    - Code None.

13. *Signal_conditioner_2*

    - A unique identifier for an additional signal conditioner.
    - Code None.

14. *Signal_conditioner_3*

    - A unique identifier for an additional signal conditioner.
    - Code None.

15. *Signal_conditioner_4*

    - A unique identifier for an additional signal conditioner.
    - Code None.

16. *Corrected_sample_rate*

    - Sample rate associated with the corrected data, in nanoseconds (e.g. 2ns per point).
    - Code None.

17. *Acquisition_instrumentation*

    - Indicates the type of data acquisition method that was used to capture the data.
Code_string	Description
01	One Tek 7912
02	Two Tek 7912s
03	Three Tek 7912s
04	Four Tek 7912s

10	200ms LeCroy
11	One Tek 7912 and one 200ms LeCroy
20	Network analyzer
30	Peak Level Detector (PLR)

18. *Shot_time*
    - Time of shot in military time (e.g.: 1400 is equivalent to 2:00 pm).
    - Code None.
19. *Shot_date*
    - Date of shot in format DD-MMM-YY (e.g.: 12-FEB-85).
    - Code None.
20. *Environmental_level*
    - Peak pulser amplitude for the shot, in Volts/Meter.
    - Code None.
21. *Reference_rise_time*
    - Pulser rise time as measured by free field sensor.
    - Code None.
22. *Total_raw_data_points*
    - Total number of data points in the time-tied/uncorrected data record.
    - Code None.
23. *Total_corrected_data_points*
    - Total number of data points in the corrected data record.
    - Code None.
24. *Raw_peak_amplitude*
    - Time domain peak absolute value from uncorrected data record stored in AIRBASE in units indicated under measurement_units.

- Code None.

25. *Corrected_peak_amplitude*

    - Time domain peak absolute value from corrected data record stored in AIRBASE in units indicated under measurement_units.
    - Code None.

26. *Corrected_impulse*

    - Time integral of corrected waveform.
    - Code None.

27. *Corrected_action*

    - Time integral of squared corrected waveform.
    - Code None.

28. *Corrected_rectified*

    - Time integral of absolute value of corrected waveform.
    - Code None.

29. *Corrected_impulse_peak*

    - Time integral of corrected waveform maximized with respect to upper time limit.
    - Code None.

30. *Corrected_max_rise*

    - Maximum rate of change between adjacent time points for the corrected waveform.
    - Code None.

31. *Measurement_remarks*

    - Includes any comments on the data record.
    - Code None.

32. *Corrected_measurement_record*

    - Time domain record of corrected data in compressed format.
    - Code None.

Bibliography

[1] J.T. Robinson, "Airbase Reference Manual", TRW Report no. 52375-6004-UT-82, August 1990.

[2] "Macintosh System 7 Reference Manual", Apple Computer, Inc., 1991

[3] M. Renner, "4th DIMENSION Design Reference Manual",Acius, Inc. 1989

[4] W. Mayall, "4th DIMENSION Language Reference Manual", Acius, Inc. 1989

[5] D. Brandt, "4th DIMENSION User Reference Manual",Acius, Inc. 1989

[6] M. Renner, "4th DIMENSION Tutorials", Acius, Inc. 1989

[7] "MATLAB User's Guide", The Mathworks, Inc., May 15, 1991.

[8] "MATLAB Signal Processing Toolbox", The Mathworks, Inc., Aug. 29, 1988.

[9] K. A. Sawires, "EMP Development, Analysis and Testing EXTRAP Users Manual, Version 1.0", TRW Report no. 51594-6005-UT-00, Prepared for Air Force Weapons Laboratory, February 1990.

[10] B. Parruck, and S. M. Riad, "An optimization criterion for iterative deconvolution." *IEEE Transactions on Instrumentation and Measurement*, vol. IM-32, pp. 137-140, March 1983.

[11] "Posttest Analysis for EMP Facilities/Test Operations Improvements", TRW Report no. 41452-6002-UT-00, Prepared for Air Force Weapons Labs, March 1983.

DISTRIBUTION LIST

DNA-TR-92-105

DEPARTMENT OF DEFENSE

DEFENSE SPECIAL WEAPONS AGENCY
ATTN: ESA
ATTN: ESE, MIKE RODNEY
ATTN: EST, R GULLICKSON
2 CY ATTN: TRC
ATTN: WE

DEFENSE TECHNICAL INFORMATION CENTER
2 CY ATTN: DTIC/OCP

DEPARTMENT OF DEFENSE CONTRACTORS

JAYCOR
2 CY ATTN: DR CYRUS P KNOWLES

KAMAN SCIENCES CORPORATION
ATTN: DASIAC
ATTN: DASIAC/DARE

TRW S. I. G.
2 CY ATTN: KALIM A SAWIRES