Research Note 86-07

THE COMPUTER-AIDED ANALYTIC PROCESS MODEL: APPENDIX TO THE OPERATIONS HANDBOOK FOR THE APM DEMONSTRATION PACKAGE

Ronald G. Shapiro
Dunlap and Associates, East, Incorporated

for

ARI FIELD UNIT AT FORT BENNING, GEORGIA
Joel D. Schendel, Acting Chief

TRAINING RESEARCH LABORATORY
Seward Smith, Acting Director

U. S. Army
Research Institute for the Behavioral and Social Sciences

January 1986

U. S. ARMY RESEARCH INSTITUTE

FOR THE BEHAVIORAL AND SOCIAL SCIENCES

A Field Operating Agency under the Jurisdiction of the

Deputy Chief of Staff for Personnel

Research accomplished under contract
for the Department of the Army

Dunlap and Associates, East, Incorporated

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER Research Note 86-07	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) The Computer-Aided Analytic Process Model: Appendix to the Operations Handbook for the APM Demonstration Package		5. TYPE OF REPORT & PERIOD COVERED Final Report May 1980 - February 1983
		6. PERFORMING ORG. REPORT NUMBER 293-26
7. AUTHOR(s) Ronald G. Shapiro		8. CONTRACT OR GRANT NUMBER(s) MDA903-80-C-0345
9. PERFORMING ORGANIZATION NAME AND ADDRESS Dunlap and Associates East, Inc. 17 Washington Street Norwalk, CT 06584		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 2Q263743A794
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Research Institute for the Behavioral and Social Sciences 5001 Eisenhower Ave., Alexandria, VA 22333-5600		12. REPORT DATE January 1986
		13. NUMBER OF PAGES 318
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) ARI Field Unit P.O. Box 2086 Fort Benning, Georgia 31905		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Dr. Seward Smith, Contracting Officer's Representative

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Computer-Aided Model; Analytic Process Model; PASCAL; Computer Program; Apple

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Computer-Aided APM Demonstration Package provides the analyst with the opportunity to perform a thorough analysis of a system while the computer keeps track of the analysis and insures that the analyst examines the parts of the data base which are of interest. This is, however, a demonstration package which can only process small data bases. Because the package is implemented on an Apple II Plus, processing is relatively slow. An explanation of the APM, listings of the data sets derived using the APM and recommendations for further development of the APM appear in the companion volume--"The Analytic Process Model for System

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. Design and Measurement: A Computer-Aided Tool for Analyzing Training Systems and Other Human-Machine Systems." A separate companion volume—"The Computer-Aided Analytic Process Model: Operations Handbook for the APM Demonstation Package" is also available under separate cover. ~~The~~ present volume, which is an Appendix to the Operations Handbook, contains the actual PASCAL computer code listings. Disks containing this code and the data bases in machine-readable format are also available. Keywords: [illegible] computer program; [illegible] computer program; PERFITEM ([illegible] Program), MEASATTR (Measures [illegible] Program); MEASATR computer program; MEASPROP (Measures-Process Program), [illegible] program.

Accession For
NTIS GRA&I ✓
DTIC TAB ☐
Unannounced ☐
Justification

By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

QUALITY INSPECTED 3

TABLE OF CONTENTS

I. EXECUTIVE SUMMARY

This report is a supporting document to the Final Summary Report* on an analytic process model (APM) for systems design and measurement. The present document, the Appendix to the Operations Manual for the computer-aided version of the APM,** contains a listing of the Pascal code for the computer-aided model. The model was developed for the Army Research Institute (ARI) Field Unit, Fort Benning, over the period from March 1980 to February 1983.

The objective of the computer-aided APM is to provide a routinized, thorough, adaptive and efficient procedure to help testers, analysts and researchers develop design specifications and evaluation measures for any planned or existing human-machine system, and especially for any training system. The demonstration version of the computer-aided model, as described in this report, performs a sample of the routines expected in any ultimate version that may be developed in the future. Specifically, the demonstration model helps one to derive evaluation measures, but not design specifications. In addition, it contains data bases for training systems, but not for any other human-machine system. Finally, it contains data bases for only half of the six training subsystems (for design, enabling and delivery, but not for command, logistics or emplacement). For demonstration purposes, this development represents an appropriate and sufficient allocation of project resources, since the more significant effort was needed to develop the underlying concepts for both a feasible "manual" model and the computer-aided model. The demonstration model, using an Apple II Plus computer with two 5¼-inch disk drives, programmed in PASCAL, can be exercised straight through, beginning with identifying the system and ending with a subset of its performance measures. Any larger capability than presently exists in the demonstration routine would require a computer with substantially greater capacity and speed.

Program listings are contained in Chapters 2-8. Chapter 2 contains a listing of the GREETING program which displays the title page, instructions and the analytic procedure menu. When the computer is turned on and the APM system disk is inserted, the title page is displayed first. Whenever the analyst decides to select a different analytic procedure, this program is loaded and the analytic procedure menu is displayed.

*Bloom, R.F., Oates, J.F., Jr., Shapiro, R.G. and Hamilton, J.W. The Analytic Process Model for System Design and Measurement: A Computer-Aided Tool for Analyzing Training Systems and Other Human-Machine Systems. Norwalk, CT: Dunlap and Associates East, Inc., 28 February 1983. (Final Summary Report)

**Shapiro, R.G., Bloom, R.F. and Oates, J.F., Jr. The Analytic Process Model For System Design and Measurement: Operations Handbook for the APM Demonstration Package. Norwalk, CT: Dunlap and Associates East, Inc., 28 February 1983.

Chapter 3 contains a listing of the PERFormance ITEM (PERFITEM) program. This program allows the analyst to add, reword, remove and print performance items (objectives, functional purposes and characteristics). Chapter 4 contains a listing of the MEASures and ATTRibutes (MEASATTR) program. MEASATTR allows the analyst to add, reword, remove and print attributes and measures for a given performance item. Chapter 5 contains a listing of the MEASurement PURPose (MEASPURP) program. MEASPURP allows the analyst to define a measurement purpose and associate (or disassociate) each of the characteristics with the measurement purpose.

Chapter 6 contains a listing of the PRINT program which allows the analyst to print the performance items, attributes and measures for a given measurement purpose, or an entire subsystem. Chapter 7 contains listings of the PACK program which arranges the data set for a given subsystem in order by item reference number, and packs the data sets so that any unused space is placed at the end of the data set so that it can be used.

Chapter 8 contains listings for a variety of programs which support the APM system. STARTUP asks the analyst to place the APM SYSTEM disk in Drive #1 at the appropriate time. GREETSHORT reminds the analyst to place the APM system disk in Drive #1 if he does not do so. BLOCKHELP and BLOCKINSTR set up the HELP, BRIEFHELP and INSTR data sets so that they are blocked efficiently for usage by the APM system. VIDPATCH modifies the SYSTEM.APPLE program for use with the VIDEX board. It only needs to be run once with each copy of the SYSTEM.APPLE program.

The operations handbook contains item-by-item directions for starting up and carrying out all the steps in the demonstration routine, schematic flow charts and miscellaneous information about the equipment and maintenance. Thus, it ought to be understood prior to reading the actual Pascal listings.

GREETING PROGRAM

The greeting program presents the title page, instructions (if desired), establishes which system class, system and subsystem the analyst intends to use (while allowing for the possibility of creating new ones). The Greeting Program concludes by determining which analytic procedure is to be performed next. Whenever any analytic procedure is completed, the analytic procedure menu in this program is displayed to find out which program ought to be executed next.

```
 1    1:D    1 (*$L PRINTER:*)
 1    1:D    1 (*$S+*)
 1    1:D    1 (*Program to greet user, instruct user, and set up table of systems and subsystems*)
 1    1:D    1 (*Ronald G. Shapiro      V2.0     10/25/82*)
 1    1:D    1
 1    1:D    1 Program Greeting;
 1    1:D    3
28    1:D    3
28    2:D    1   PROCEDURE SETCHAIN(TYTLE:STRING);
28    3:D    1   PROCEDURE SETCVAL(VAL:STRING);
28    4:D    1   PROCEDURE GETCVAL(VAR VAL:STRING);
28    5:D    1   PROCEDURE SWAPON;
28    6:D    1   PROCEDURE SWAPOFF;
28    6:D    1
 1    1:D    1 USES CHAINSTUFF;
 1    1:D    3
```

These procedures are part of the Apple Computer's CHAINSTUFF library entry. The demonstration package uses only SETCHAIN which causes another program to be activated.

```
1    1:D     3 (*$P*)TYPE
1    1:D     3   PASSFILE=RECORD
1    1:D     3     CURSYS,CURSP,CURSUB,PAC:STRING[80];
1    1:D     3     NCURSYS,NCURSP,NCURSUB,NPAC,FLAG1,FLAG2,FLAG3:INTEGER;
1    1:D     3     END;
1    1:D     3
1    1:D     3   SUBSYSFILE=RECORD
1    1:D     3     NSUBSYS: INTEGER;
1    1:D     3     SUBSYS:STRING[80];
1    1:D     3     END;
1    1:D     3
1    1:D     3   SPSYSFILE=RECORD
1    1:D     3     NSPSYS: INTEGER;
1    1:D     3     SPSYS:STRING[80];
1    1:D     3     END;
1    1:D     3
1    1:D     3   SYSFILE=RECORD
1    1:D     3     NSYSTEM: INTEGER;
1    1:D     3     SYSTEM:STRING[80];
1    1:D     3     END;
1    1:D     3
1    1:D     3   INSTRFILE=RECORD
1    1:D     3     LINE:ARRAY[1..20] OF STRING[80];
1    1:D     3     END;
1    1:D     3
1    1:D     3   HELPFILE=RECORD
1    1:D     3     LINE:ARRAY[1..10] OF STRING[80];
1    1:D     3     END;
1    1:D     3
1    1:D     3   FASTFILE=RECORD
1    1:D     3     PRINTIT:ARRAY[1..300]OF BOOLEAN;
1    1:D     3     END;
1    1:D     3
```

PASSFILE passes information about: 1) system class [CURSYS,NCURSYS] 2) system [CURSP,NCURSP] 3) subsystem [CURSB,NCURSB] 4) aspect [PAC,NPAC] from one program to another. Flag 1 is used to tell the GREETING program whether to begin with title page or analytic procedure list. Flags 2 and 3 are unused. SUBSYSFILE contains a list of the defined subsystems for each system. SPFILE contains a list of the defined systems for each system class. SYSFILE contains a list of the defined system classes. INSTRFILE contains the instructions. HELPFILE contains the help commands. FASTFILE allows fast printing of a measurement purpose if the measurement purpose had been printed before.

```
   1     1:D      3 (*$P*)VAR
   1     1:D      3   XFUNPUR,XOBJECTIVE,PAC,CURSYS,CURSP,CURSUB,LINE,REGLINE,ANSWER:STRING[80];
   1     1:D    372   ANSHOLD,ANS2,ANS:CHAR;
   1     1:D    375   DONE,OK,OVER,NEG:BOOLEAN;
   1     1:D    379   NLENGTH,LLENGTH,PGE,I,NDATA,II,II2,J,K,L,M,N,NFUNPUR,NOBJECTIVE,
   1     1:D    379     NPAC,NCURSYS,NCURSP,NCURSUB:INTEGER;
   1     1:D    397   JHELP,HELP:INTEGER;
   1     1:D    399   CORELAST,EII:INTEGER[8];
   1     1:D    405   APMDSK:STRING[8];
   1     1:D    410   NAMEFILETEST,NAMEFASTISSUE,FILESPNAME,FRAME:STRING[24];
   1     1:D    462
   1     1:D    462   ASPECT:ARRAY[1..5] OF STRING[14];
   1     1:D    502   SUBSYS,SPSYS,SYSTEM:ARRAY[1..10] OF STRING[80];
   1     1:D   1732   SCRATCH:ARRAY[1..20]OF STRING[80];
   1     1:D   2552   NSCRATCH:ARRAY[1..20] OF INTEGER;
   1     1:D   2572   NSUBSYS,NSPSYS,NSYSTEM:ARRAY[1..10] OF INTEGER;
   1     1:D   2602
   1     1:D   2602   SYSLIST:FILE OF SYSFILE;
   1     1:D   2944   SUBSYSLIST:FILE OF SUBSYSFILE;
   1     1:D   3286   SPSYSLIST:FILE OF SPSYSFILE;
   1     1:D   3628   PASSNODE:FILE OF PASSFILE;
   1     1:D   4099   INSTFILE:FILE OF INSTRFILE;
   1     1:D   5219   HELPER:FILE OF HELPFILE;
   1     1:D   5929   PRNT:TEXT;
   1     1:D   6230   FILETEST:TEXT;
   1     1:D   6531   FASTISSUE:FILE OF FASTFILE;
   1     1:D   7131
```

These strings, arrays and variables are used by the GREETING program.

```
1     2:D     1 (*$P*)PROCEDURE KEY;FORWARD;
1     3:D     1 PROCEDURE KEYN;FORWARD;
1     4:D     1 PROCEDURE BRANCHOUT;FORWARD;
1     5:D     1 PROCEDURE SYSTEMFILES;FORWARD;
1     6:D     1 PROCEDURE S1;FORWARD;
1     7:D     1 PROCEDURE S2;FORWARD;
1     8:D     1 PROCEDURE S5;FORWARD;
1     9:D     1 PROCEDURE MENU;FORWARD;
1    10:D     1 PROCEDURE PROPERMAINDISK;FORWARD;
1    11:D     1 PROCEDURE OPENSPFILES;FORWARD;
1    12:D     1 PROCEDURE GOSPSYSCREATE;FORWARD;
1    13:D     1 PROCEDURE SPSYSCREATE;FORWARD;
1    14:D     1 PROCEDURE SPSYSTEMFILES;FORWARD;
1    15:D     1 PROCEDURE SUBSYSTEMFILES;FORWARD;
1    16:D     1 PROCEDURE PREPSPCREATE;FORWARD;
1    17:D     1 PROCEDURE HELPROUTINE;FORWARD;
1    18:D     1 PROCEDURE GOSUBCREATE;FORWARD;
1    18:D     1
1    18:D     1
```

These procedures are presented later on in the GREETING program.

```
 1   19:D     1 (*$P*)PROCEDURE KEYNPREP(HLP:INTEGER;MSG:STRING);
 1   19:0     0   BEGIN
 1   19:1     0     HELP:=HLP;
 1   19:1     9     WRITE(MSG);
 1   19:1    20     KEYN;
 1   19:0    22     END;
 1   19:0    34
```

KEYNPREP displays a one line message, then calls KEYN to read a number from the keyboard.

```
1   20:D     1 (*$P*)PROCEDURE PREPKEY(HLP:INTEGER;MSG:STRING);
1   20:0     0   BEGIN
1   20:1     0     HELP:=HLP;
1   20:1     9     REPEAT
1   20:2     9       WRITE(MSG);
1   20:2    20       KEY;
1   20:1    22       UNTIL (ANS='Y') OR (ANS='N');
1   20:0    35     END;
1   20:0    50
```

PREPKEY displays a message then calls KEY to read a letter response from the keyboard. If a response is not Y, y, N, n, Yes or No, it redisplays the message and, once again, waits for a response.

```
 1     2:D     1 (*$P*)PROCEDURE KEY;
 1     2:0     0   BEGIN
 1     2:0     0     (*$R-*)
 1     2:1     0     ANSWER:='                    ';
 1     2:1    24     REPEAT
 1     2:2    24       READLN(ANSWER);
 1     2:2    43       ANS:=ANSWER[1];
 1     2:2    50       IF (ANS<>'Y')AND(ANS<>'N')AND(ANS<>'H')AND(ANS<>'y')AND
 1     2:2    73         (ANS<>'n')AND(ANS<>'h') THEN
 1     2:3    87         WRITELN('PLEASE RESPOND YES OR NO!');
 1     2:2   132       IF ORD(ANS)>90 THEN
 1     2:3   139         BEGIN
 1     2:4   139           II2:=ORD(ANS)-32;
 1     2:4   147           ANS:=CHR(II2);
 1     2:3   153           END;
 1     2:1   153       UNTIL (ANS='Y') OR (ANS='N') OR (ANS='H');
 1     2:1   172       (*$R+*)
 1     2:1   172       IF ANS='H' THEN
 1     2:2   179         HELPROUTINE;
 1     2:0   181       END;
 1     2:0   196
```

KEY reads a letter response from the keyboard. If response is 1) y or Y, it places a Y in ANS and returns to calling procedure; 2) n or N, it places an N in ANS and returns to calling procedure; 3) h or H, it calls the HELP routine, places an H in ANS and returns to calling program; or 4) any other key--it displays PLEASE RESPOND YES OR NO and awaits a Y, N, H, y, n or h response. NOTE: Only the first character/line is processed. The rest is ignored.

```
 1   21:D     1 (*$P*)PROCEDURE ANYKEY;
 1   21:0     0   BEGIN
 1   21:1     0     WRITELN(' ');
 1   21:1    18     WRITELN('*** Please press any key to continue ***');
 1   21:1    78     (*$R-*)
 1   21:1    78     READ(ANS);
 1   21:1    89     (*$R+*)
 1   21:0    89     END;
 1   21:0   102
```

ANYKEY displays "Please Press any Key to Continue" then it awaits a Keypress before returning control to the calling procedure.

```
1    3:D     1 (*$P*)PROCEDURE KEYN;
1    3:D     1   VAR
1    3:D     1     ANSWER: STRING[40];
1    3:D    22     II: ARRAY[1..4] OF INTEGER;
1    3:D    26     OK:BOOLEAN;
1    3:D    27      II0:INTEGER;
1    3:D    28
1    3:0     0   BEGIN
1    3:0     0     (*$R-*)
1    3:1     0     OK:=TRUE;
1    3:1     3     REPEAT
1    3:2     3       REPEAT
1    3:3     3         I:=-1;
1    3:3     8         ANSWER:='                              ';
1    3:3    35         READLN(ANSWER);
1    3:3    54         IF LENGTH(ANSWER)=0 THEN
1    3:4    62           WRITELN('Please enter the integer again');
1    3:2   112         UNTIL LENGTH(ANSWER)<>0;
1    3:2   120       IF (ANSWER[1]='H') OR (ANSWER[1]='h') THEN
1    3:3   135         BEGIN
1    3:4   135           HELPROUTINE;
1    3:4   137           I:=999;
1    3:4   143           EXIT(KEYN);
1    3:3   147           END;
1    3:2   147       FOR I:=1 TO 4 DO
1    3:3   162         BEGIN
1    3:4   162           II[I]:=ORD(ANSWER[I])-48;
1    3:4   180           IF (II[I]<0) OR (II[I]>9) THEN
1    3:5   207             BEGIN
1    3:6   207               IF (I=1) OR (II[I]<>(ORD(' ')-48)) THEN
1    3:7   229                 BEGIN
1    3:8   229                   OK:=FALSE;
1    3:8   232                   WRITELN('PLEASE RESPOND WITH A POSITIVE INTEGER');
1    3:7   290                   END;
1    3:5   290               END;
1    3:3   290           END;
1    3:1   300       UNTIL TRUE;
1    3:1   303     II0:=II[1];
1    3:1   313     FOR I:=2 TO 4 DO
1    3:2   328       BEGIN
1    3:3   328         IF (II[I]>=0) AND (II[I]<=9) THEN
1    3:4   355           II0:=II0*10+II[I];
1    3:2   372         END;
1    3:2   382     (*$R+*)
1    3:1   382     I:=II0;
1    3:0   387     END;
1    3:0   410
```

KEYN reads a 1 or 2 digit response from the keyboard and places it into I. If an H or an h are typed in, it places a 999 in I and calls the HELP routine. If more than 2 characters are typed, only 2 characters are read. The rest are ignored. If the character(s) are not positive intergers, KEYN will display an appropriate warning and wait for a response.

```
1   22:D     1 (*$P*)PROCEDURE SHOWALINE;
1   22:0     0   BEGIN
1   22:1     0     NLENGTH:=LENGTH(LINE);
1   22:1     7     WHILE LINE[NLENGTH]=' ' DO
1   22:2    18       NLENGTH:=NLENGTH-1;
1   22:1    28     IF NLENGTH<=LLENGTH THEN
1   22:2    37       BEGIN
1   22:3    37         WRITE(LINE);
1   22:3    48         EXIT(SHOWALINE);
1   22:2    52         END;
1   22:1    52     L:=LLENGTH;
1   22:1    58     WHILE LINE[L]<>' ' DO
1   22:2    69       L:=L-1;
1   22:1    79     L:=L-1;
1   22:1    87     REGLINE:=COPY(LINE,1,L);
1   22:1   104     L:=L+2;
1   22:1   112     WRITELN(REGLINE);
1   22:1   131     NLENGTH:=NLENGTH-L+1;
1   22:1   143     REGLINE:=COPY(LINE,L,NLENGTH);
1   22:1   162     WRITE('  ',REGLINE);
1   22:0   187     END;
1   22:0   204
1   22:0   204 (*$I #5:HELPTEXT.TEXT*)
```

SHOWALINE displays text on the screen. If, by chance, the text is longer than the amount of space available on the current line, the display continues onto a second line.

```
1   23:D     1 (*$P*)PROCEDURE PRNTHELP;
1   23:0     0   BEGIN
1   23:1     0     DONE:=FALSE;
1   23:1     4     REWRITE(PRNT,'PRINTER:');
1   23:1    25     PAGE(PRNT);
1   23:1    35     WRITELN(PRNT,CHR(14),'Analytic Process Model',CHR(13));
1   23:1    97     WRITELN(PRNT,CHR(14),'Help File',chr(13));
1   23:1   146     PGE:=2;
1   23:1   150     REPEAT
1   23:2   150       SEEK(HELPER,PGE);
1   23:2   161       GET(HELPER);
1   23:2   169       PAGE(PRNT);
1   23:2   179       K:=PGE-1;
1   23:2   187       WRITELN(PRNT,'                                                ',
1   23:2   239              '                                Page ',K);
1   23:2   308       FOR J:=1 TO 10 DO
1   23:3   322         WRITELN(PRNT,HELPER^.LINE[J]);
1   23:2   362       IF COPY(HELPER^.LINE[2],2,10)='conclusion' THEN
1   23:3   399         DONE:=TRUE;
1   23:2   403       PGE:=PGE+1;
1   23:1   411       UNTIL(DONE);
1   23:1   416     PAGE(PRNT);
1   23:1   426     CLOSE(PRNT);
1   23:0   435     END;
1   23:0   452
```

PRINTHELP prints the HELP file on the printer. It is called by HELPROUTINE.

```
1    17:D     1 (*$P*)PROCEDURE HELPROUTINE;
1    17:0     0   BEGIN
1    17:0     0     (*$I-*)
1    17:1     0     RESET(HELPER,'#5:HELP');
1    17:1    18     (*$I+*)
1    17:1    18     I:=IORESULT;
1    17:1    23     IF (I<>0) THEN
1    17:2    30       BEGIN
1    17:3    30         PAGE(OUTPUT);
1    17:3    40         WRITELN('UNFORTUNATELY, THE HELP FILE IS NOT AVAILABLE ON YOUR DISK');
1    17:3   118         WRITELN(' ');
1    17:3   136         WRITELN('PLEASE PRESS ANY KEY TO CONTINUE PROCESSING');
1    17:3   199         READ(ANS);
1    17:3   210         EXIT(HELPROUTINE);
1    17:2   214         END;
1    17:1   214       I:=0;
1    17:1   218       PGE:=HELP+1;
1    17:1   226       DONE:=FALSE;
1    17:1   230       REPEAT
1    17:2   230         SEEK(HELPER,PGE);
1    17:2   241         GET(HELPER);
1    17:2   249         PAGE(OUTPUT);
1    17:2   259         GOTOXY(73,0);
1    17:2   264         K:=PGE-1;
1    17:2   272         WRITELN('Page ',K);
1    17:2   309         GOTOXY(0,0);
1    17:2   314         FOR J:=1 TO 10 DO
1    17:3   328           WRITELN(HELPER^.LINE[J]);
1    17:2   368         IF COPY(HELPER^.LINE[2],2,10)='conclusion' THEN
1    17:3   405           DONE:=TRUE;
1    17:2   409         WRITELN(' ');
1    17:2   427         WRITELN('****PLEASE PRESS RETURN KEY TO VIEW NEXT PAGE****');
1    17:2   496         WRITELN('****PLEASE TYPE PAGE NUMBER AND PRESS RETURN KEY TO VIEW ANOTHER
                        PAGE****');
1    17:2   589         WRITE  ('****PLEASE PRESS ESC AND RETURN KEYS TO ESCAPE HELP ROUTINE****');
1    17:2   664         PGE:=PGE+1;
1    17:2   672         (*$R-*)
1    17:2   672         ANSWER:='          ';
1    17:2   689         READLN(ANSWER);
1    17:2   708         page(output);
```

HELPROUTINE displays appropriate help commands when it is called by KEY or KEYN. HELPROUTINE knows which HELP to display because the calling program places the appropriate help page number into HELP. Once the analyst sees the first help message, he/she can ask for other help messages by typing in the page number of the desired help messages. Note that the HELP file is made by editing a series of files (HELP1 . . . HELPN) using the Apple editor. Then, they are processed by the BLOCKHELP program (see Chapter VIII). The HELP file produced by BLOCKHELP is suitable for use with the HELPROUTINE. HELPROUTINE "knows" it has hit the last page of the file because the word "conclusion" appears on the second line of the last page.

```
1   17:2   718        IF ORD(ANSWER[1])=27 THEN
1   17:3   726          BEGIN
1   17:4   726            CLOSE(HELPER);
1   17:4   735            (*$R+*)
1   17:4   735            EXIT (HELPROUTINE);
1   17:4   739            (*$R-*)
1   17:3   739            END;
1   17:2   739        IF (ANSWER[1]>='0') AND (ANSWER[1]<='9') THEN
1   17:3   754          BEGIN
1   17:4   754            PGE:=ORD(ANSWER[1])-48;
1   17:4   763            IF (ANSWER[2]>='0') AND (ANSWER[2]<='9') THEN
1   17:5   778              PGE:=PGE*10 + ORD(ANSWER[2])-48;
1   17:4   793            PGE:=PGE+1;
1   17:4   801            IF PGE<2 THEN
1   17:5   808              PGE:=2;
1   17:4   812            DONE:=FALSE;
1   17:3   816            END;
1   17:1   816        UNTIL (DONE) AND ((ANSWER[1]<'0') OR (ANSWER[1]>'9'));
1   17:1   835    PAGE(OUTPUT);
1   17:1   845    PREPKEY(2,'Would you like to print the help file?');
1   17:1   889    IF ANS='Y' THEN
1   17:2   896      BEGIN
1   17:3   896        KEYNPREP(2,'How many copies? ');
1   17:3   919        FOR N:=1 TO I DO
1   17:4   935          PRNTHELP;
1   17:3   947        WRITELN('DONE');
1   17:2   971        END;
1   17:1   971    CLOSE(HELPER)
1   17:1   980    (*$R+*)
1   17:0   980    END;
1   17:0  1000
1   17:0  1000
1   17:0  1000 (*$I #5:HELPTEXT.TEXT*)
1   17:0  1000
```

See previous page for program description.

```
 1   24:D     1 (*$P*)PROCEDURE QUIT;
 1   24:0     0   BEGIN
 1   24:1     0     PAGE(OUTPUT);
 1   24:1    10     REPEAT
 1   24:2    10       write('Would you like to return to title page?');
 1   24:2    61       help:=2;
 1   24:2    65       key
 1   24:1    65       until (ans='Y') or (ans='N');
 1   24:1    80     if ans='Y' then
 1   24:2    87       begin
 1   24:3    87         setchain('greeting');
 1   24:3   101         passnode^.flag1:=0;
 1   24:3   109         branchout;
 1   24:3   111         exit(program);
 1   24:2   115         end;
 1   24:1   115     REPEAT
 1   24:2   115       write('Would you like to turn off computer for now?');
 1   24:2   171       help:=2;
 1   24:2   175       key
 1   24:1   175       until (ans='Y') or (ans='N');
 1   24:1   190     if ans='Y' then
 1   24:2   197       begin
 1   24:3   197         passnode^.flag1:=0;
 1   24:3   205         branchout;
 1   24:3   207         page(output);
 1   24:3   217         writeln('bye...');
 1   24:3   243         writeln('');
 1   24:3   263         writeln(' I hope to see you again very soon!');
 1   24:3   318         writeln(' ');
 1   24:3   336         writeln('You may now:');
 1   24:3   368         writeln(' 1. Remove the disks.');
 1   24:3   409         writeln(' 2. Turn off printer.');
 1   24:3   450         writeln(' 3. Turn off computer.');
 1   24:3   492         writeln(' 4. Turn off this display screen.');
 1   24:3   545         OVER:=FALSE;
 1   24:3   549         REPEAT
 1   24:4   549           I:=1
 1   24:3   549           UNTIL OVER=TRUE;
 1   24:2   561         end;
 1   24:1   561     REPEAT
```

QUIT asks the analyst what he/she wants to do next 1) return to the title page, 2) turn off the computer, 3) access the Apple operating system. If options 2 or 3 are selected, it says bye . . . and displays some helpful advice. If option 1 is selected, the computer then goes into an infinite loop, whereas if option 3 is selected, the analyst gains control of the Apple operating system. If the analyst selects no option, then he/she is sent back into the APM demonstration package, approximately where he/she left off.

```
349   1   24:2   561        write('Would you like to leave model and access computer operating system?');
350   1   24:2   640        help:=2;
351   1   24:2   644        key
352   1   24:1   644        until (ans='Y') or (ans='N');
353   1   24:1   659      if ans='Y' then
354   1   24:2   666        begin
355   1   24:3   666          passnode^.flag1:=0;
356   1   24:3   674          branchout;
357   1   24:3   676          PAGE(OUTPUT);
358   1   24:3   686          writeln('bye...');
359   1   24:3   712          writeln('');
360   1   24:3   732          writeln('  I hope to see you again very soon!');
361   1   24:3   788          writeln('');
362   1   24:3   808          writeln(' You are now on your own with the Apple OS');
363   1   24:3   870          writeln('');
364   1   24:3   890          writeln(' good luck...');
365   1   24:3   923          for i:=1 to 1000 do
366   1   24:4   939            I:=I*10;
367   1   24:3   957          exit(program);
368   1   24:2   961          end;
369   1   24:1   961      writeln('Since you have selected not to exit from this program, I will ');
370   1   24:1  1043      writeln(' send you back to where you left off as soon as you press any key.');
371   1   24:1  1129      (*$r-*)
372   1   24:1  1129      read(ans);
373   1   24:1  1140      (*$r+*)
374   1   24:0  1140      END;
375   1   24:0  1166
```

See previous page for program description.

```
 1   25:D     1 (*$P*)PROCEDURE HELLO;
 1   25:0     0   BEGIN
 1   25:1     0     PAGE(OUTPUT);
 1   25:1    10     WRITELN(' ');
 1   25:1    28     WRITELN(' ');
 1   25:1    46     WRITELN(' ');
 1   25:1    64     WRITELN(' ');
 1   25:1    82     WRITELN('                          AN ANALYTIC PROCESS MODEL FOR');
 1   25:1   156     WRITELN(' ');
 1   25:1   174     WRITELN('                         SYSTEMS DESIGN AND MEASUREMENT:');
 1   25:1   249     WRITELN(' ');
 1   25:1   267     WRITELN('                         APPLICATIONS TO TRAINING SYSTEMS');
 1   25:1   343     WRITELN(' ');
 1   25:1   361     WRITELN(' ');
 1   25:1   379     WRITELN(' ');
 1   25:1   397     WRITELN(' ');
 1   25:1   415     WRITELN(' ');
 1   25:1   433     WRITELN('     Prepared for: ARI Field Unit, Fort Benning, Georgia');
 1   25:1   509     WRITELN('     Prepared by: Dunlap & Associates East, Inc., Norwalk, Conn');
 1   25:1   599     WRITELN('     Date: 25 October 1982');
 1   25:1   645     WRITELN(' ');
 1   25:1   663     WRITELN(' ');
 1   25:1   681     WRITE('                         PLEASE PRESS ANY KEY TO BEGIN');
 1   25:1   746     (*$R-*)
 1   25:1   746     READ(ANS);
 1   25:1   757     (*$R+*)
 1   25:0   757     END;
 1   25:0   770
 1   25:0   770
```

HELLO displays the title page.

```
 1   26:D     1 (*$P*)PROCEDURE PRNTINSTRUCTIONS;
 1   26:0     0   BEGIN
 1   26:1     0     DONE:=FALSE;
 1   26:1     4     REWRITE(PRNT,'PRINTER:');
 1   26:1    25     PAGE(PRNT);
 1   26:1    35     WRITELN(PRNT,CHR(14),'Analytic Process Model',CHR(13));
 1   26:1    97     WRITELN(PRNT,CHR(14),'Instructions',chr(13));
 1   26:1   149     PGE:=2;
 1   26:1   153     REPEAT
 1   26:2   153       SEEK(INSTFILE,PGE);
 1   26:2   164       GET(INSTFILE);
 1   26:2   172       PAGE(PRNT);
 1   26:2   182       K:=PGE-1;
 1   26:2   190       WRITELN(PRNT,'                                                ',
 1   26:2   242               '                                   Page ',K);
 1   26:2   311       FOR J:=1 TO 20 DO
 1   26:3   325         WRITELN(PRNT,INSTFILE^.LINE[J]);
 1   26:2   365       IF COPY(INSTFILE^.LINE[2],2,10)='conclusion' THEN
 1   26:3   402         DONE:=TRUE;
 1   26:2   406       PGE:=PGE+1;
 1   26:1   414       UNTIL(DONE);
 1   26:1   419     PAGE(PRNT);
 1   26:1   429     CLOSE(PRNT);
 1   26:0   438     END;
 1   26:0   454
```

PRINTINSTRUCTIONS prints the instructions on the printer when it is called by INSTRUCTIONS. (It is nearly identical in structure to PRINTHELP.)

```
1   27:D     1 (*$P*)PROCEDURE INSTRUCTIONS;
1   27:0     0   BEGIN
1   27:0     0     (*$I-*)
1   27:1     0     RESET(INSTFILE,'APMUTL:INSTRUCT');
1   27:1    26     (*$I+*)
1   27:1    26     I:=IORESULT;
1   27:1    31     IF I=9 THEN
1   27:2    38       BEGIN
1   27:3    38         PROPERMAINDISK;
1   27:3    40         INSTRUCTIONS;
1   27:3    42         EXIT(INSTRUCTIONS);
1   27:2    46         END;
1   27:1    46     IF (I<>0)AND(I<>9) THEN
1   27:2    59       BEGIN
1   27:3    59         PAGE(OUTPUT);
1   27:3    69         WRITELN('UNFORTUNATELY, INSTRUCTION FILE IS NOT AVAILABLE ON YOUR DISK');
1   27:3   154         WRITELN(' ');
1   27:3   172         WRITELN('PLEASE PRESS ANY KEY TO CONTINUE PROCESSING');
1   27:3   235         READ(ANS);
1   27:3   246         EXIT(INSTRUCTIONS);
1   27:2   250         END;
1   27:1   250       I:=0;
1   27:1   254       PGE:=2;
1   27:1   258       DONE:=FALSE;
1   27:1   262       REPEAT
1   27:2   262         SEEK(INSTFILE,PGE);
1   27:2   273         GET(INSTFILE);
1   27:2   281         PAGE(OUTPUT);
1   27:2   291         GOTOXY(73,0);
1   27:2   296         K:=PGE-1;
1   27:2   304         WRITELN('Page ',K);
1   27:2   341         GOTOXY(0,0);
1   27:2   346         FOR J:=1 TO 20 DO
1   27:3   360           WRITELN(INSTFILE^.LINE[J]);
1   27:2   400         IF COPY(INSTFILE^.LINE[2],2,10)='conclusion' THEN
1   27:3   437           DONE:=TRUE;
1   27:2   441         WRITELN(' ');
1   27:2   459         WRITELN('****PLEASE PRESS RETURN KEY TO VIEW NEXT PAGE****');
1   27:2   528         WRITELN('****PLEASE TYPE PAGE NUMBER AND PRESS RETURN KEY TO VIEW ANOTHER
                       PAGE****');
```

INSTRUCTIONS displays the instructions. Functionally, it is virtually identical to HELPROUTINE.

```
1   27:2   621       WRITE  ('****PLEASE PRESS ESC AND RETURN KEYS TO ESCAPE INSTRUCTIONS****');
1   27:2   696       PGE:=PGE+1;
1   27:2   704       (*$R-*)
1   27:2   704       ANSWER:='          ';
1   27:2   721       READLN(ANSWER);
1   27:2   740       page(output);
1   27:2   750       IF ORD(ANSWER[1])=27 THEN
1   27:3   758         BEGIN
1   27:4   758           CLOSE(INSTFILE);
1   27:4   767           (*$R+*)
1   27:4   767           EXIT (INSTRUCTIONS);
1   27:4   771           (*$R-*)
1   27:3   771           END;
1   27:2   771       IF (ANSWER[1]>='0') AND (ANSWER[1]<='9') THEN
1   27:3   786         BEGIN
1   27:4   786           PGE:=ORD(ANSWER[1])-48;
1   27:4   795           IF (ANSWER[2]>='0') AND (ANSWER[2]<='9') THEN
1   27:5   810             PGE:=PGE*10 + ORD(ANSWER[2])-48;
1   27:4   825           PGE:=PGE+1;
1   27:4   833           IF PGE<2 THEN
1   27:5   840             PGE:=2;
1   27:4   844           DONE:=FALSE;
1   27:3   848           END;
1   27:1   848       UNTIL (DONE) AND ((ANSWER[1]<'0') OR (ANSWER[1]>'9'));
1   27:1   867   PAGE(OUTPUT);
1   27:1   877   PREPKEY(2,'Would you like to print these instructions?');
1   27:1   926   IF ANS='Y' THEN
1   27:2   933     BEGIN
1   27:3   933       KEYNPREP(2,'How many copies? ');
1   27:3   956       FOR N:=1 TO I DO
1   27:4   972         PRNTINSTRUCTIONS;
1   27:3   984       WRITELN('DONE');
1   27:2  1008       END;
1   27:1  1008   CLOSE(INSTFILE)
1   27:1  1017   (*$R+*)
1   27:0  1017   END;
1   27:0  1038
```

See previous page for program description.

```
 1    4:D      1 (*$P*)PROCEDURE BRANCHOUT;
 1    4:0      0  BEGIN
 1    4:0      0    (*$I-*)
 1    4:1      0    RESET(PASSNODE,'PASSTHRU');
 1    4:1     19    (*$I+*)
 1    4:1     19    IF(IORESULT<>0) THEN
 1    4:2     25      WRITE('SERIOUS ERROR -- NO FILE PASSTHRU AT BRANCHOUT')
 1    4:1     83    ELSE
 1    4:2     85      BEGIN
 1    4:3     85        PASSNODE^.CURSYS:=CURSYS;
 1    4:3     95        PASSNODE^.CURSUB:=CURSUB;
 1    4:3    104        PASSNODE^.CURSP:=CURSP;
 1    4:3    114        PASSNODE^.PAC:=PAC;
 1    4:3    122        PASSNODE^.NCURSYS:=NCURSYS;
 1    4:3    132        PASSNODE^.NCURSUB:=NCURSUB;
 1    4:3    142        PASSNODE^.NCURSP:=NCURSP;
 1    4:3    152        PASSNODE^.NPAC:=NPAC;
 1    4:3    162        PASSNODE^.FLAG1:=0;
 1    4:3    170        PUT(PASSNODE);
 1    4:3    178        IF EOF(PASSNODE) THEN
 1    4:4    188          WRITELN('OUT OF DISK SPACE WHILE WRITING PASSTHRU');
 1    4:3    248        CLOSE(PASSNODE,LOCK);
 1    4:2    257        END;
 1    4:0    257    END;
 1    4:0    272
```

BRANCHOUT loads the PASSTHRU file with appropriate data for use by called programs.

```
  1   28:D     1 (*$P*)PROCEDURE BRANCHIN;
  1   28:0     0  BEGIN
  1   28:0     0    (*$I-*)
  1   28:1     0    RESET(PASSNODE,'PASSTHRU');
  1   28:1    19    (*$I+*)
  1   28:1    19    IF IORESULT<>0 THEN
  1   28:2    25      BEGIN
  1   28:3    25        REWRITE(PASSNODE,'PASSTHRU');
  1   28:3    46        PASSNODE^.CURSYS:='';
  1   28:3    56        PASSNODE^.CURSUB:='';
  1   28:3    66        PASSNODE^.PAC:='';
  1   28:3    74        PASSNODE^.NCURSYS:=0;
  1   28:3    82        PASSNODE^.NCURSP:=0;
  1   28:3    90        PASSNODE^.NCURSUB:=0;
  1   28:3    98        PASSNODE^.NPAC:=0;
  1   28:3   106        PUT(PASSNODE);
  1   28:3   114        IF EOF(PASSNODE) THEN
  1   28:4   124          WRITELN('OUT OF DISK SPACE WHILE WRITING PASSTHRU');
  1   28:3   184        CLOSE(PASSNODE,LOCK);
  1   28:3   193        RESET(PASSNODE,'PASSTHRU')
  1   28:2   214        END;
  1   28:1   214    GET(PASSNODE);
  1   28:1   222    CURSYS:=PASSNODE^.CURSYS;
  1   28:1   232    CURSP:=PASSNODE^.CURSP;
  1   28:1   242    CURSUB:=PASSNODE^.CURSUB;
  1   28:1   251    PAC:=PASSNODE^.PAC;
  1   28:1   259    NCURSYS:=PASSNODE^.NCURSYS;
  1   28:1   268    NCURSUB:=PASSNODE^.NCURSUB;
  1   28:1   277    NPAC:=PASSNODE^.NPAC;
  1   28:1   286    CLOSE(PASSNODE,LOCK);
  1   28:0   295    END;
  1   28:0   310
```

BRANCHIN gets information from the PASSTHRU file for use by this program.

```
1  29:D    1  (*$P*)PROCEDURE MAKEDISK;
1  29:0    0    BEGIN
1  29:1    0      PREFKEY(54,'Would you like to prepare a new data disk for this subsystem?');
1  29:1   67      IF ANS='N' THEN
1  29:2   74        BEGIN
1  29:3   74          MENU;
1  29:3   76          EXIT(MAKEDISK);
1  29:2   80          END;
1  29:1   80      WRITELN('I can neither format nor name a disk, so I will tell you what to do',
1  29:1  159              chr(13),'  then you will have the opportunity to do it.',chr(13),
1  29:1  237              '  1. Turn to page 184 of the APPLE PASCAL Operating System Reference
                                Manual.', chr(13),
1  29:1  335              '  2. Insert Apple 3 in drive 1, and follow the directions.',chr(13),
1  29:1  415              '  3. When finished, turn to page 33 of the same manual.',chr(13),
1  29:1  492              '  4. Enter the FILER program.',chr(13),
1  29:1  543              '  5. Turn to page 45 and run the change program, changing',chr(13),
1  29:1  622              '       BLANK: to the new name followed by a colon.  The new',chr(13),
1  29:1  703              '       name consists of the first 3 letters of the system name',chr(13),
1  29:1  787              '       followed by the first 3 letters of the subsystem name.',chr(13),
1  29:1  870              '  6. Good Luck--I will see you again soon');
1  29:1  931       EXIT(PROGRAM);
1  29:0  935       END;
1  29:0  948
```

MAKEDISK tells analyst how to format a new disk if no disk is available for the subsystem requested.

```
1   10:D     1  (*$P*)PROCEDURE PROPERMAINDISK;
1   10:0     0    BEGIN
1   10:1     0      PAGE(OUTPUT);
1   10:1    10      REPEAT
1   10:1    10        (*$I-*)
1   10:2    10        RESET(INSTFILE,'APMUTL:INSTRUCT');
1   10:2    36        (*$I+*)
1   10:2    36        K:=IORESULT;
1   10:2    41        IF K=0 THEN
1   10:3    48          CLOSE(INSTFILE);
1   10:2    57        IF K=9 THEN
1   10:3    64          BEGIN
1   10:4    64            WRITELN('Please place the APM UTILity disk (APMUTIL) in drive # 2.');
1   10:4   141            ANYKEY;
1   10:3   143            END;
1   10:1   143        UNTIL K<>9;
1   10:0   150    END;
1   10:0   164
```

PROPERMAINDISK checks to be sure that the APMUTILITY disk is in Drive #2 when it is needed.

```
1   30:D     1  (*$P*)PROCEDURE PROPERDISK;
1   30:0     0   BEGIN
1   30:1     0     OK:=TRUE;
1   30:1     4     PAGE(OUTPUT);
1   30:1    14     WRITELN('System class: ',CURSYS);
1   30:1    60     WRITELN('System: ',CURSP);
1   30:1   100     WRITELN('Subsystem: ',CURSUB);
1   30:1   142     WRITELN(' ');
1   30:1   160     APMDSK:=CONCAT(COPY(CURSYS,1,2),COPY(CURSP,1,2),COPY(CURSUB,1,2),':');
1   30:1   245     NAMEFILETEST:=CONCAT(APMDSK,'TEST');
1   30:1   277     REPEAT
1   30:1   277       (*$I-*)
1   30:2   277       RESET(FILETEST,NAMEFILETEST);
1   30:2   288       (*$I+*)
1   30:2   288       K:=IORESULT;
1   30:2   293       IF K=9 THEN
1   30:3   300         BEGIN
1   30:4   300           HELP:=2;
1   30:4   304           WRITELN('If the disk for this system class, system, and subsystem is',
1   30:4   375                   chr(13),'  available, place it in Drive # 2 and type Y (retrn).',
1   30:4   459                   chr(13),'  Otherwise, type N (return).');
1   30:4   518           KEY;
1   30:4   520           IF ANS='Y' THEN
1   30:5   527             BEGIN
1   30:6   527               PROPERDISK;
1   30:6   529               EXIT(PROPERDISK);
1   30:5   533               END;
1   30:4   533           IF (ANS='N') OR (ANS='n') THEN
1   30:5   546             OK:=FALSE;
1   30:3   550           END;
1   30:1   550       UNTIL (K<>9) OR (OK=FALSE);
1   30:1   564     IF OK=FALSE THEN
1   30:2   572       MAKEDISK;
1   30:0   574     END;
1   30:0   590
```

PROPERDISK checks to be sure that the appropriate disk for the system class, system and subsystem selected is in Drive #2 before branching to another program.

```
1 31:D   1 (*$P*)PROCEDURE REMOVEFASTISSUE;
1 31:0   0   BEGIN
1 31:0   0     (*$I-*)
1 31:1   0     NAMEFASTISSUE:=CONCAT(APMDSK,COPY(CURSYS,1,4),COPY(CURSP,1,4),COPY(CURSUB,1,4),'FA');
1 31:1  95     RESET(FASTISSUE,NAMEFASTISSUE);
1 31:1 106     CLOSE(FASTISSUE,PURGE);
1 31:1 113     (*$I+*)
1 31:0 113     END;
1 31:0 126
```

REMOVEFASTISSUE delets the FASTISSUE file whenever PACKDATA is run. [A new FASTISSUE file will be created the next time the PRINT program is run.]

```
1 32:D   1 (*$P*)PROCEDURE SHOWMENU;
1 32:0   0   BEGIN
1 32:1   0     page(output);
1 32:1  10     WRITELN('System class: ',cursys);
1 32:1  56     WRITELN('System: ',cursp);
1 32:1  96     WRITELN('Subsystem: ',cursub);
1 32:1 138     WRITELN(' ');
1 32:1 156     WRITELN('You may perform the following analytic procedures:');
1 32:1 226     writeln(' ');
1 32:1 244     writeln(  '1.  Add, modify, or delete performance items',chr(13),
1 32:1 310               '2.  Add, modify, or delete measurable attributes or measures',chr(13),
1 32:1 392               '3.  Add, modify, or delete measurement purposes',chr(13),
1 32:1 461               '4.  Print out selected results from your analysis',chr(13),
1 32:1 532               '5.  Pack your disk files most efficiently (a slow process)',chr(13),
1 32:1 612               '6.  Change System class, System, and/or Subsystem to be analyzed',chr(13),
1 32:1 698               '7.  Review Instructions',chr(13),
1 32:1 743               '8.  Stop for now',chr(13),chr(13));
1 32:0 799     END;
1 32:0 812
```

SHOWMENU displays the list of analytic procedures available.

```
 1    9:D     1 (*$P*)PROCEDURE MENU;
 1    9:0     0   BEGIN
 1    9:1     0     SHOWMENU;
 1    9:1     2     REPEAT
 1    9:2     2       KEYNPREP(5,'Which would you like to do?');
 1    9:2    35       IF I=999 THEN
 1    9:3    44         BEGIN
 1    9:4    44           MENU;
 1    9:4    46           EXIT(MENU);
 1    9:3    50           END;
 1    9:2    50       IF (I>8) OR (I<1) THEN
 1    9:3    63         WRITELN('Please type an integer between 1 and 8');
 1    9:1   121       UNTIL (I>0) AND (I<9);
 1    9:1   134     IF I<6 THEN
 1    9:2   141         BEGIN
 1    9:3   141           PROPERDISK;
 1    9:3   143           IF K=9 THEN
 1    9:4   150             BEGIN
 1    9:5   150               WRITELN('Options 1 to 5 are not available, because you are not able',
 1    9:5   220                       chr(13), 'to insert the appropriate disk.  Please select',
 1    9:5   288                       chr(13), 'Option 6, 7, or 8 when the menu reappears');
 1    9:5   359               ANYKEY;
 1    9:5   361               MENU;
 1    9:4   363               END;
 1    9:2   363           END;
 1    9:2   363
 1    9:1   363     CASE I OF
 1    9:1   368       1: BEGIN
 1    9:3   368            SETCHAIN('PERFITEM');
 1    9:3   382            EXIT(PROGRAM);
 1    9:2   386            END;
 1    9:1   388       2: BEGIN
 1    9:3   388            SETCHAIN('MEASATTR');
 1    9:3   402            EXIT(PROGRAM);
 1    9:2   406            END;
 1    9:1   408       3: BEGIN
 1    9:3   408            SETCHAIN('MEASPURP');
 1    9:3   422            EXIT(PROGRAM);
 1    9:2   426            END;
 1    9:1   428       4: BEGIN
```

MENU calls SHOWMENU to display the list of analytic procedures available. Then, menu calls KEYN to find out which analytic procedure the analyst wishes to perform. It then sets up the Apple chaining program to execute the desired procedure. Then, it branches to that procedure.

```
 1    9:3   428          SETCHAIN('PRINT');
 1    9:3   439          EXIT(PROGRAM);
 1    9:2   443          END;
 1    9:1   445      5: BEGIN
 1    9:3   445          REMOVEFASTISSUE;
 1    9:3   447          SETCHAIN('PACKDATA');
 1    9:3   461          EXIT(PROGRAM);
 1    9:2   465          END;
 1    9:1   467      6: BEGIN
 1    9:3   467          PROPERMAINDISK;
 1    9:3   469          SYSTEMFILES;
 1    9:3   471          SPSYSTEMFILES;
 1    9:3   473          SUBSYSTEMFILES;
 1    9:3   475          BRANCHOUT;
 1    9:3   477          MENU;
 1    9:3   479          EXIT(MENU);
 1    9:2   483          END;
 1    9:1   485      7: BEGIN
 1    9:3   485          PROPERMAINDISK;
 1    9:3   487          INSTRUCTIONS;
 1    9:3   489          MENU;
 1    9:2   491          END;
 1    9:1   493      8: BEGIN
 1    9:3   493          QUIT;
 1    9:3   495          MENU;
 1    9:3   497          EXIT(MENU);
 1    9:2   501          END;
 1    9:1   503      END;
 1    9:0   526    END;
 1    9:0   548
```

See previous page for program description.

```
1 33:D   1 (*$P*)PROCEDURE SUBSYSCREATE;
1 33:0   0   BEGIN
1 33:1   0     REPEAT
1 33:2   0       I:=0;
1 33:2   4       REPEAT
1 33:3   4         I:=I+1;
1 33:2  12         UNTIL(I=10) OR (SUBSYS[I]='');
1 33:2  38       IF I=10 THEN
1 33:3  45         BEGIN
1 33:4  45           WRITELN('***WARNING SYST: NO ROOM FOR MORE SUBSYSTEMS FOR SYSTEM CLASS',CURSYS);
1 33:4 138           ANYKEY;
1 33:4 140           EXIT(SUBSYSCREATE)
1 33:3 144           END
1 33:2 144         ELSE
1 33:3 146           GOSUBCREATE;
1 33:1 148       UNTIL OK;
1 33:0 153   END;
1 33:0 170
```

SUBSYSCREATE enters subsystem names into the SUBSYSFILE for a given system class and system.

```
1 18:D    1(*$P*)PROCEDURE GOSUBCREATE;
1 18:0    0  BEGIN
1 18:1    0    WRITE('What is the name of your subsystem?');
1 18:1   47    SUBSYS[I]:='';
1 18:1   65    REPEAT
1 18:2   65      READLN(SUBSYS[I]);
1 18:2   95      IF SUBSYS[I]='' THEN
1 18:3  115        EXIT(SUBSYSCREATE);
1 18:2  119      IF LENGTH(SUBSYS[I])<5 THEN
1 18:3  138        WRITE('Subsystem name must contain at least 5 letters--',CHR(13),
1 18:3  208              'Please type a new name:');
1 18:2  243        K:=POS(' ',SUBSYS[I]);
1 18:2  268      IF (K>0) AND (K<6) THEN
1 18:3  281        WRITE('None of the first five characters of subsystem name can be blank--',chr(13),
1 18:3  369              'Please type a new name:');
1 18:1  404      UNTIL (LENGTH(SUBSYS[I])>=5) AND ((K<1) OR (K>5));
1 18:1  435    NSUBSYS[I]:=I;
1 18:1  452    WRITELN('Subsystem ',SUBSYS[I],' is member number ',NSUBSYS[I],
1 18:1  549            CHR(13),' of system ',CURSP);
1 18:1  602    RESET(SUBSYSLIST,FRAME);
1 18:1  615    SEEK(SUBSYSLIST,I);
1 18:1  626    SUBSYSLIST^.NSUBSYS:=NSUBSYS[I];
1 18:1  644    SUBSYSLIST^.SUBSYS:=SUBSYS[I];
1 18:1  664    PUT (SUBSYSLIST);
1 18:1  672    CLOSE(SUBSYSLIST,LOCK);
1 18:1  681    WRITELN(' ');
1 18:1  699    REPEAT
1 18:2  699      WRITELN('Would you like to proceed with the analysis of system class ',CURSYS,
1 18:2  783              ';',CHR(13),'system ',CURSP,' ;subsystem ',SUBSYS[I],'?');
1 18:2  898      HELP:=33;
1 18:2  902      KEY;
1 18:1  904      UNTIL (ANS='Y') OR (ANS='N');
1 18:1  917     IF ANS='Y' THEN
1 18:2  924       BEGIN
1 18:3  924         CURSUB:=SUBSYS[I];
1 18:3  941         NCURSUB:=NSUBSYS[I];
1 18:3  958         PASSNODE^.FLAG1:=0;
1 18:3  966         BRANCHOUT;
1 18:3  968         MENU;
1 18:2  970         END;
```

GOSUBCREATE is a continuation of SUBSYSCREATE.

```
794   1   18:1   970     REPEAT
795   1   18:2   970       WRITE('Would you like to add more subsystems to system ',CURSP,'?');
796   1   18:2  1052       HELP:=54;
797   1   18:2  1056       KEY
798   1   18:1  1056       UNTIL (ANS='Y') OR (ANS='N');
799   1   18:1  1071     OK:=TRUE;
800   1   18:1  1075     IF ANS='Y' THEN
801   1   18:2  1082       OK:=FALSE
802   1   18:1  1082     ELSE
803   1   18:2  1088       EXIT(SUBSYSCREATE);
804   1   18:0  1092   END;
805   1   18:0  1110
805   1   18:0  1110 (*$I #5:GREET2.TEXT*)
```

See previous page for program description.

```
 1   34:D     1 (*$P*)PROCEDURE OPENSYSTEMFILES;
 1   34:0     0   BEGIN
 1   34:1     0     PROPERMAINDISK;
 1   34:1     2     (*$I-*)
 1   34:1     2     RESET(SYSLIST,'APMUTL:APMSYSTEMS');
 1   34:1    30     (*$I+*)
 1   34:1    30     IF IORESULT<>0 THEN
 1   34:2    36       BEGIN
 1   34:3    36         REWRITE(SYSLIST,'APMUTL:APMSYSTEMS');
 1   34:3    66         FOR I:=1 TO 10 DO
 1   34:4    80           BEGIN
 1   34:5    80             SYSLIST^.NSYSTEM:=I;
 1   34:5    87             SYSLIST^.SYSTEM:='';
 1   34:5    97             SEEK(SYSLIST,I);
 1   34:5   108             PUT(SYSLIST);
 1   34:5   116             IF EOF(SYSLIST) THEN
 1   34:6   126               BEGIN
 1   34:7   126                 WRITELN('OUT OF DISK SPACE!!!');
 1   34:7   166                 ANYKEY;
 1   34:7   168                 EXIT(SYSTEMFILES);
 1   34:6   172                 END;
 1   34:4   172             END;
 1   34:3   182         CLOSE(SYSLIST,LOCK);
 1   34:3   191         OPENSYSTEMFILES;
 1   34:3   193         EXIT(OPENSYSTEMFILES);
 1   34:2   197         END;
 1   34:1   197       BEGIN
 1   34:2   197         FOR I:=1 TO 10 DO
 1   34:3   211           BEGIN
 1   34:4   211             SEEK(SYSLIST,I);
 1   34:4   222             GET(SYSLIST);
 1   34:4   230             NSYSTEM[I]:=SYSLIST^.NSYSTEM;
 1   34:4   248             SYSTEM[I]:=SYSLIST^.SYSTEM;
 1   34:3   268             END;
 1   34:2   278         CLOSE(SYSLIST,LOCK);
 1   34:1   287         END;
 1   34:0   287       END;
 1   34:0   306
```

OPENSYSTEMFILES opens the file containing the list of defined system classes. If such a file does not exist, GOSYSTEMFILES creates one.

```
1   35:D     1 (*$P*)PROCEDURE OPENSUBFILES;
1   35:0     0   BEGIN
1   35:0     0     (*$I-*)
1   35:1     0     RESET(SUBSYSLIST,FRAME);
1   35:1    11     (*$I+*)
1   35:1    11     IF IORESULT<>0 THEN
1   35:2    17       BEGIN
1   35:3    17         REWRITE(SUBSYSLIST,FRAME);
1   35:3    30         FOR I:=1 TO 10 DO
1   35:4    44           BEGIN
1   35:5    44             SUBSYSLIST^.NSUBSYS:=I;
1   35:5    51             SUBSYSLIST^.SUBSYS:='';
1   35:5    61             SEEK(SUBSYSLIST,I);
1   35:5    72             PUT(SUBSYSLIST);
1   35:5    80             IF EOF(SUBSYSLIST) THEN
1   35:6    90               BEGIN
1   35:7    90                 WRITELN('OUT OF DISK SPACE!!!');
1   35:7   130                 ANYKEY;
1   35:7   132                 EXIT(OPENSUBFILES);
1   35:6   136                 END;
1   35:4   136             END;
1   35:3   146         CLOSE(SUBSYSLIST,LOCK);
1   35:3   155         RESET(SUBSYSLIST,FRAME);
1   35:2   168         END;
1   35:1   168       BEGIN
1   35:2   168         FOR I:=1 TO 10 DO
1   35:3   182           BEGIN
1   35:4   182             SEEK(SUBSYSLIST,I);
1   35:4   193             GET(SUBSYSLIST);
1   35:4   201             NSUBSYS[I]:=SUBSYSLIST^.NSUBSYS;
1   35:4   219             SUBSYS[I]:=SUBSYSLIST^.SUBSYS;
1   35:3   239             END;
1   35:2   249         CLOSE(SUBSYSLIST,LOCK);
1   35:1   258         END;
1   35:0   258       END;
1   35:0   276
1   35:0   276
```

OPENSUBFILES opens file containing the names of the subsystems for a given system class and system. If such a file does not exist for the given system class and system, it creates it.

```
  1   36:D     1 (*$P*)PROCEDURE PREPSUBCREATE;
  1   36:0     0   BEGIN
  1   36:1     0     FRAME:=CONCAT('APMUTL:',COPY(CURSYS,1,5),COPY(CURSP,1,5),'SUB');
  1   36:1    82     OPENSUBFILES;
  1   36:1    84     SUBSYSCREATE;
  1   36:0    86     END;
  1   36:0    98
```

PREPSUBCREATE calls OPENSUBFILES and SUBSYSCREATE as necessary.

```
1   37:D     1 (*$P*)PROCEDURE SYSCREATE;
1   37:0     0   BEGIN
1   37:1     0     REPEAT
1   37:2     0       I:=0;
1   37:2     4       REPEAT
1   37:3     4         I:=I+1;
1   37:2    12         UNTIL (I=10) OR (SYSTEM[I]='');
1   37:2    38       IF I=10 THEN
1   37:3    45         BEGIN
1   37:4    45           WRITELN('***WARNING SYSTEM IS FULL***');
1   37:4    93           ANYKEY;
1   37:4    95           EXIT(SYSCREATE);
1   37:3    99           END
1   37:2    99         ELSE
1   37:3   101           BEGIN
1   37:4   101             WRITE('What is the name of your new class of systems?');
1   37:4   159             SYSTEM[I]:='';
1   37:4   177             REPEAT
1   37:5   177               READLN(SYSTEM[I]);
1   37:5   207               IF SYSTEM[I]='' THEN
1   37:6   227                 EXIT(SYSCREATE);
1   37:5   231               IF LENGTH(SYSTEM[I])<5 THEN
1   37:6   250                 WRITE('System class name must contain at least 5 characters',
1   37:6   314                       CHR(13),'Please type a new name:');
1   37:5   359               K:=POS(' ',SYSTEM[I]);
1   37:5   384               IF (K>0) AND (K<6) THEN
1   37:6   397                 WRITELN('None of the first five characters of a system class name
                               can be blank--',chr(13),
1   37:6   490                          'Please type a new name');
1   37:4   532               UNTIL(LENGTH(SYSTEM[I])>=5) AND ((K<1) OR (K>5));
1   37:4   563             NSYSTEM[I]:=I;
1   37:4   580             WRITELN('System class ',SYSTEM[I],' has been added to the list of
                            system classes');
1   37:4   692             WRITELN('  as system number ',NSYSTEM[I]);
1   37:4   754             RESET(SYSLIST,'APMUTL:APMSYSTEMS');
1   37:4   784             SEEK(SYSLIST,I);
1   37:4   795             SYSLIST^.NSYSTEM:=NSYSTEM[I];
1   37:4   813             SYSLIST^.SYSTEM:=SYSTEM[I];
1   37:4   833             PUT (SYSLIST);
1   37:4   841             CLOSE(SYSLIST,LOCK);
```

SYSCREATE enters system class names into the list of system classes.

```
926   1   37:4    850            CURSYS:=SYSTEM[I];
927   1   37:4    868            NCURSYS:=NSYSTEM[I];
928   1   37:4    885            WRITELN(' ');
929   1   37:4    903            REPEAT
930   1   37:5    903              WRITE('Would you like to define new systems for system class  ',
                                   SYSTEM[I],'?');
931   1   37:5   1002              HELP:=56;
932   1   37:5   1006              KEY
933   1   37:4   1006              UNTIL (ANS='Y') OR (ANS='N');
934   1   37:4   1021            IF ANS='Y' THEN
935   1   37:5   1028              BEGIN
936   1   37:6   1028                PREPSPCREATE;
937   1   37:5   1030                END
938   1   37:4   1030              ELSE
939   1   37:5   1032                S2;
940   1   37:4   1034              PREPKEY(33,'Would you like to develop a new class of systems?');
941   1   37:4   1089              IF ANS='Y' THEN
942   1   37:5   1096                OK:=FALSE
943   1   37:4   1096              ELSE
944   1   37:5   1102                EXIT(SYSCREATE);
945   1   37:3   1106            END
946   1   37:1   1106        UNTIL OK;
947   1   37:0   1111  END;
948   1   37:0   1136
```

See previous page for program description.

```
1    5:D     1 (*$P*)PROCEDURE SYSTEMFILES;
1    5:0     0   BEGIN
1    5:1     0     ANSHOLD:=' ';
1    5:1     4     OPENSYSTEMFILES;
1    5:1     6     REPEAT
1    5:2     6       OVER:=TRUE;
1    5:2    10       PAGE(OUTPUT);
1    5:2    20       NDATA:=0;
1    5:2    24       WRITELN(' I have data for the following classes of human-machine systems:');
1    5:2   108       FOR I:=1 TO 10 DO
1    5:3   122         BEGIN
1    5:4   122           IF SYSTEM[I]<>'' THEN
1    5:5   142             BEGIN
1    5:6   142               WRITELN('  ',NSYSTEM[I],'.  ',SYSTEM[I]);
1    5:6   224               NDATA:=1
1    5:5   224               END;
1    5:3   228           END;
1    5:2   238       WRITELN(' ');
1    5:2   256       IF NDATA= 0 THEN
1    5:3   263         BEGIN
1    5:4   263           I:=0;
1    5:4   267           WRITELN('  ... none');
1    5:3   297           END
1    5:2   297         ELSE
1    5:3   299           BEGIN
1    5:4   299             REPEAT
1    5:5   299               KEYNPREP(33,'Which system class would you like to analyze (type 0
                             for none of the above)?');
1    5:5   381                 IF I=999 THEN
1    5:6   390                   BEGIN
1    5:7   390                     SYSTEMFILES;
1    5:7   392                     EXIT(SYSTEMFILES);
1    5:6   396                     END;
1    5:5   396                 IF(I<0) OR (I>10) THEN
1    5:6   409                   WRITELN('PLEASE TYPE AN INTEGER BETWEEN 0 AND 10')
1    5:4   468               UNTIL (I>=0) AND (I<=10);
1    5:4   481               IF I<>0 THEN
1    5:5   488                 BEGIN
1    5:6   488                   IF SYSTEM[I]='' THEN
1    5:7   508                     BEGIN
```

SYSTEMFILES displays the names of defined system classes and determines which one the analyst wishes to analyze.

```
1   5:8   508                  WRITELN(NSYSTEM[I],' DOES NOT EXIST AT PRESENT');
1   5:8   577                  WRITELN('PLEASE TRY ANOTHER SYSTEM CLASS');
1   5:8   628                  ANYKEY;
1   5:8   630                  OVER:=FALSE;
1   5:7   634                  END;
1   5:5   634                END;
1   5:3   634          END;
1   5:2   634      IF I=0 THEN
1   5:3   641        BEGIN
1   5:4   641          PREPKEY(33,'Would you like to develop a new class of systems?');
1   5:4   696          IF ANS='Y' THEN
1   5:5   703            BEGIN
1   5:6   703              SYSCREATE;
1   5:6   705              OVER:=FALSE
1   5:5   705              END
1   5:4   709            ELSE
1   5:5   711              BEGIN
1   5:6   711                PREPKEY(2,'Would you like to stop for now?');
1   5:6   748                IF ANS='Y' THEN
1   5:7   755                  QUIT
1   5:6   755                  ELSE
1   5:7   759                    BEGIN
1   5:8   759                      WRITELN('There are no other options--so I will present
                                   the options again');
1   5:8   842                      WRITELN('***Please press any key to review the options***');
1   5:8   910                      (*$R-*)
1   5:8   910                      READ(ANS);
1   5:8   921                      (*$R+*)
1   5:7   921                      END;
1   5:5   921                END;
1   5:4   921            OVER:=FALSE;
1   5:3   925            END;
1   5:1   925      UNTIL OVER;
1   5:1   930  CURSYS:=SYSTEM[I];
1   5:1   948  NCURSYS:=NSYSTEM[I];
1   5:0   965  END;
1   5:0   994
```

See previous page for program description.

```
1   15:D     1 (*$P*)PROCEDURE SUBSYSTEMFILES;
1   15:0     0   BEGIN
1   15:1     0     FRAME:=CONCAT('APMUTL:',COPY(CURSYS,1,5),COPY(CURSP,1,5),'SUB');
1   15:1    82     OPENSUBFILES;
1   15:1    84     REPEAT
1   15:2    84       OVER:=TRUE;
1   15:2    88       PAGE(OUTPUT);
1   15:2    98       LINE:=CURSP;
1   15:2   105       LLENGTH:=27;
1   15:2   109       WRITE('I have data for the following subsystems of system: ');
1   15:2   173       SHOWALINE;
1   15:2   175       WRITELN(' ');
1   15:2   193       NDATA:=0;
1   15:2   197       FOR I:=1 TO 10 DO
1   15:3   211         BEGIN
1   15:4   211           IF SUBSYS[I]<>'' THEN
1   15:5   231            BEGIN
1   15:6   231             WRITELN('   ',NSUBSYS[I],'.  ',SUBSYS[I]);
1   15:6   313             NDATA:=1;
1   15:5   317             END;
1   15:3   317           END;
1   15:2   327       WRITELN(' ');
1   15:2   345       IF NDATA=0 THEN
1   15:3   352         BEGIN
1   15:4   352           WRITELN('   ... none');
1   15:4   382           S1
1   15:3   382           END
1   15:2   384         ELSE
1   15:3   386           REPEAT
1   15:4   386             REPEAT
1   15:5   386               KEYNPREP(54,'Which subsystem would you like to analyze (type 0 for
                             none of the above)?');
1   15:5   465               IF I=999 THEN
1   15:6   474                 BEGIN
1   15:7   474                   SUBSYSTEMFILES;
1   15:7   476                   EXIT(SUBSYSTEMFILES);
1   15:6   480                   END;
1   15:5   480               IF(I<0) OR (I>10) THEN
1   15:6   493                 WRITELN('PLEASE TYPE AN INTEGER BETWEEN 0 AND 10')
1   15:4   552               UNTIL (I>=0)AND (I<=10);
```

SUBSYSTEMFILES displays the names of the defined subsystems for a given system.

```
1   15:4   565           OK:=FALSE;
1   15:4   569           IF I<>0 THEN
1   15:5   576             BEGIN
1   15:6   576               OK:=TRUE;
1   15:6   580               IF SUBSYS[I]='' THEN
1   15:7   600                 BEGIN
1   15:8   600                   WRITELN(NSUBSYS[I],' DOES NOT EXIST AT PRESENT');
1   15:8   669                   WRITELN('PLEASE TRY ANOTHER SYSTEM');
1   15:8   714                   OK:=FALSE
1   15:7   714                   END;
1   15:5   718               END
1   15:4   718             ELSE
1   15:5   720               BEGIN
1   15:6   720                 OK:=TRUE;
1   15:6   724                 OVER:=FALSE;
1   15:5   728                 END
1   15:3   728           UNTIL OK;
1   15:2   733       IF I=0 THEN
1   15:3   740         SI;
1   15:1   742     UNTIL OVER;
1   15:1   747   CURSUB:=SUBSYS[I];
1   15:1   764   NCURSUB:=NSUBSYS[I];
1   15:0   781   END;
1   15:0   806
```

See previous page for program description.

```
1087 1 6:D   1(*$P*)PROCEDURE S1;
1088 1 6:0   0  BEGIN
1089 1 6:1   0    PREPKEY(54,'Would you like to add subsystems to this system?');
1090 1 6:1  54    IF (ANS='Y') OR (ANS='y') THEN
1091 1 6:2  67      BEGIN
1092 1 6:3  67        SUBSYSCREATE;
1093 1 6:3  69        OVER:=FALSE
1094 1 6:2  69        END
1095 1 6:1  73      ELSE
1096 1 6:2  75        BEGIN
1097 1 6:3  75          PREPKEY(33,'Would you like to process another class of systems?');
1098 1 6:3 132          IF ANS='Y' THEN
1099 1 6:4 139            BEGIN
1100 1 6:5 139              SYSTEMFILES;
1101 1 6:5 141              SPSYSTEMFILES;
1102 1 6:5 143              SUBSYSTEMFILES;
1103 1 6:5 145              PASSNODE^.FLAG1:=0;
1104 1 6:5 153              BRANCHOUT;
1105 1 6:5 155              MENU;
1106 1 6:4 157              END;
1107 1 6:3 157          PREPKEY(2,'Would you like to stop for now?');
1108 1 6:3 194          IF (ANS='Y') OR (ANS='y') THEN
1109 1 6:4 207            QUIT
1110 1 6:3 207            ELSE
1111 1 6:4 211              BEGIN
1112 1 6:5 211                WRITELN('THERE ARE NO MORE OPTIONS--SO I WILL PRESENT OPTIONS LIST AGAIN');
1113 1 6:5 294                WRITELN('***Please press any key to continue***');
1114 1 6:5 352                (*$R-*)
1115 1 6:5 352                READ(ANS);
1116 1 6:5 363                (*$R+*)
1117 1 6:4 363                END;
1118 1 6:4 363
1119 1 6:3 363        OVER:=FALSE;
1120 1 6:2 367      END;
1121 1 6:0 367    END;
1122 1 6:0 384
```

S1 is a continuation of SUBSYSTEMFILES.

```
1123 1 7:D    1 (*$P*)PROCEDURE S2;
1124 1 7:0    0   BEGIN
1125 1 7:1    0     REPEAT
1126 1 7:2    0       WRITE('You have chosen not to divide system class ',CURSYS,' into systems',
1127 1 7:2   92             chr(13),'Would you like to proceed with applying the model to this system?');
1128 1 7:2  179       HELP:=33;
1129 1 7:2  183       KEY
1130 1 7:1  183       UNTIL (ANS='Y') OR (ANS='N');
1131 1 7:1  198     IF ANS='Y' THEN
1132 1 7:2  205       BEGIN
1133 1 7:3  205         FILESPNAME:=CONCAT('APMUTL:',COPY(CURSYS,1,5),'SP');
1134 1 7:3  262         OPENSPFILES;
1135 1 7:3  264         RESET(SPSYSLIST,FILESPNAME);
1136 1 7:3  277         SEEK(SPSYSLIST,1);
1137 1 7:3  286         SPSYSLIST^.NSPSYS:=1;
1138 1 7:3  291         SPSYSLIST^.SPSYS:=CURSYS;
1139 1 7:3  301         PUT(SPSYSLIST);
1140 1 7:3  309         CLOSE(SPSYSLIST,LOCK);
1141 1 7:3  318         CURSP:=CURSYS;
1142 1 7:3  326         NCURSP:=1;
1143 1 7:3  330         FRAME:=CONCAT('APMUTL:',COPY(CURSYS,1,5),COPY(CURSP,1,5),'SUB');
1144 1 7:3  412         OPENSUBFILES;
1145 1 7:3  414         RESET(SUBSYSLIST,FRAME);
1146 1 7:3  427         SEEK(SUBSYSLIST,1);
1147 1 7:3  436         SUBSYSLIST^.NSUBSYS:=1;
1148 1 7:3  441         SUBSYSLIST^.SUBSYS:=CURSYS;
1149 1 7:3  451         PUT (SUBSYSLIST);
1150 1 7:3  459         CLOSE(SUBSYSLIST,LOCK);
1151 1 7:3  468         CURSUB:=CURSYS;
1152 1 7:3  475         NCURSUB:=1;
1153 1 7:3  479         PASSNODE^.FLAG1:=0;
1154 1 7:3  487         BRANCHOUT;
1155 1 7:3  489         MENU;
1156 1 7:2  491         END;
1157 1 7:0  491     END;
1158 1 7:0  508
1159 1 7:0  508
1160 1 7:0  508 (*$I #5:GREET2.TEXT*)
1161 1 7:0  508
1161 1 7:0  508 (*$I #5:GREET3.TEXT*)
```

S2 is a continuation of SYSCREATE.

```
1   11:0     1 (*$P*)PROCEDURE OPENSPFILES;
1   11:0     0   BEGIN
1   11:0     0     (*$I-*)
1   11:1     0     RESET(SPSYSLIST,FILESPNAME);
1   11:1    11     (*$I+*)
1   11:1    11     IF IORESULT<>0 THEN
1   11:2    17       BEGIN
1   11:3    17         REWRITE(SPSYSLIST,FILESPNAME);
1   11:3    30         FOR I:=1 TO 10 DO
1   11:4    44           BEGIN
1   11:5    44             SPSYSLIST^.NSPSYS:=I;
1   11:5    51             SPSYSLIST^.SPSYS:='';
1   11:5    61             SEEK(SPSYSLIST,I);
1   11:5    72             PUT(SPSYSLIST);
1   11:5    80             IF EOF(SPSYSLIST) THEN
1   11:6    90               BEGIN
1   11:7    90                 WRITELN('OUT OF DISK SPACE!!!');
1   11:7   130                 ANYKEY;
1   11:7   132                 EXIT(OPENSPFILES);
1   11:6   136                 END;
1   11:4   136             END;
1   11:3   146         CLOSE(SPSYSLIST,LOCK);
1   11:3   155         RESET(SPSYSLIST,FILESPNAME);
1   11:2   168         END;
1   11:1   168       BEGIN
1   11:2   168         FOR I:=1 TO 10 DO
1   11:3   182           BEGIN
1   11:4   182             SEEK(SPSYSLIST,I);
1   11:4   193             GET(SPSYSLIST);
1   11:4   201             NSPSYS[I]:=SPSYSLIST^.NSPSYS;
1   11:4   219             SPSYS[I]:=SPSYSLIST^.SPSYS;
1   11:3   239             END;
1   11:2   249         CLOSE(SPSYSLIST,LOCK);
1   11:1   258         END;
1   11:0   258       END;
1   11:0   276
```

OPENSPFILES opens file containing the names of all systems for a particular system class. If such a file does not exist, it creates one.

```
1 13:D   1 (*$P*)PROCEDURE SPSYSCREATE;
1 13:0   0   BEGIN
1 13:1   0     REPEAT
1 13:2   0       I:=0;
1 13:2   4       REPEAT
1 13:3   4         I:=I+1;
1 13:2  12         UNTIL(I=10) OR (SPSYS[I]='');
1 13:2  38       IF I=10 THEN
1 13:3  45         BEGIN
1 13:4  45           WRITELN('***WARNING SYST: NO ROOM FOR MORE SPSYSTEMS FOR SYSTEM CLASS',CURSYS);
1 13:4 137           ANYKEY;
1 13:4 139           EXIT(SPSYSCREATE)
1 13:3 143           END
1 13:2 143         ELSE
1 13:3 145           GOSPSYSCREATE;
1 13:1 147       UNTIL OK;
1 13:0 152   END;
1 13:0 168
```

SPSYSCREATE enters new system names into the file of system names for a particular system class.

```
1 12:D    1(*$P*)PROCEDURE GOSPSYSCREATE;
1 12:0    0  BEGIN
1 12:1    0    WRITE('What is the name of your system?');
1 12:1   44    SPSYS[I]:='';
1 12:1   62    REPEAT
1 12:2   62      READLN(SPSYS[I]);
1 12:2   92      IF SPSYS[I]='' THEN
1 12:3  112        EXIT(SPSYSCREATE);
1 12:2  116      IF LENGTH(SPSYS[I])<5 THEN
1 12:3  135        WRITE('System name must contain at least 5 letters--',CHR(13),
1 12:3  202              'Please type a new name:');
1 12:2  237        K:=POS(' ',SPSYS[I]);
1 12:2  262      IF (K>0) AND (K<6) THEN
1 12:3  275        WRITE('None of the first five characters of subsystem name can be blank--',chr(13),
1 12:3  363              'Please type a new name:');
1 12:1  398      UNTIL (LENGTH(SPSYS[I])>=5) AND ((K<1) OR (K>5));
1 12:1  429    NSPSYS[I]:=I;
1 12:1  446    WRITELN('System ',SPSYS[I],' is member number ',NSPSYS[I],chr(13),
1 12:1  550            ' of system class ',CURSYS);
1 12:1  599    RESET(SPSYSLIST,FILESPNAME);
1 12:1  612    SEEK(SPSYSLIST,I);
1 12:1  623    SPSYSLIST^.NSPSYS:=NSPSYS[I];
1 12:1  641    SPSYSLIST^.SPSYS:=SPSYS[I];
1 12:1  661    PUT (SPSYSLIST);
1 12:1  669    CLOSE(SPSYSLIST,LOCK);
1 12:1  678    WRITELN(' ');
1 12:1  696    REPEAT
1 12:2  696      WRITE  ('Would you like to proceed with the analysis of system class ',
1 12:2  768             '  ',CURSYS,',',CHR(13),'  system ',SPSYS[I],'?');
1 12:2  867      HELP:=33;
1 12:2  871      KEY
1 12:1  871      UNTIL (ANS='Y') OR (ANS='N');
1 12:1  886    IF ANS='Y' THEN
1 12:2  893      BEGIN
1 12:3  893        CURSP:=SPSYS[I];
1 12:3  911        NCURSP:=NSPSYS[I];
1 12:3  928        SUBSYSTEMFILES;
1 12:3  930        PASSNODE^.FLAG1:=0;
1 12:3  938        BRANCHOUT;
1 12:3  940        MENU;
```

GOSPSYSCREATE is a continuation of SPSYSCREATE.

```
    1   12:2   942         END;
    1   12:1   942     REPEAT
    1   12:2   942       WRITE('Would you like to add more systems to system class ',CURSYS,'?');
    1   12:2  1027       HELP:=56;
    1   12:2  1031       KEY
    1   12:1  1031       UNTIL (ANS='Y') OR (ANS='N');
    1   12:1  1046     OK:=TRUE;
    1   12:1  1050     IF ANS='Y' THEN
    1   12:2  1057       OK:=FALSE
    1   12:1  1057     ELSE
    1   12:2  1063       EXIT(SPSYSCREATE);
    1   12:0  1067   END;
    1   12:0  1086
```

See previous page for program description.

```
 1   16:D     1 (**P*)PROCEDURE PREPSPCREATE;
 1   16:0     0   BEGIN
 1   16:1     0     FILESPNAME:=CONCAT('APMUTL:',COPY(CURSYS,1,5),'SP');
 1   16:1    57     OPENSPFILES;
 1   16:1    59     SPSYSCREATE;
 1   16:0    61     END;
 1   16:0    74
```

PREPSPCREATE calls OPENSPFILES and SPSYSCREATE as necessary.

```
1   14:D     1 (*$P*)PROCEDURE SPSYSTEMFILES;
1   14:0     0   BEGIN
1   14:1     0     FILESPNAME:=CONCAT('APMUTL:',COPY(CURSYS,1,5),'SP');
1   14:1    57     OPENSPFILES;
1   14:1    59     REPEAT
1   14:2    59       OVER:=TRUE;
1   14:2    63       PAGE(OUTPUT);
1   14:2    73       WRITELN('I have data for the following systems of system class: ',CURSYS);
1   14:2   160       NDATA:=0;
1   14:2   164       FOR I:=1 TO 10 DO
1   14:3   178         BEGIN
1   14:4   178           IF SPSYS[I]<>'' THEN
1   14:5   198            BEGIN
1   14:6   198             WRITELN('  ',NSPSYS[I],'.  ',SPSYS[I]);
1   14:6   280             NDATA:=1;
1   14:5   284             END;
1   14:3   284           END;
1   14:2   294       WRITELN(' ');
1   14:2   312       IF NDATA=0 THEN
1   14:3   319         BEGIN
1   14:4   319           WRITELN('  ... none');
1   14:4   349           SS
1   14:3   349           END
1   14:2   351         ELSE
1   14:3   353           REPEAT
1   14:4   353             REPEAT
1   14:5   353               KEYNPREP(57,'Which system would you like to analyze (type 0 for
                             none of the above)?');
1   14:5   429               IF I=999 THEN
1   14:6   438                 BEGIN
1   14:7   438                   SPSYSTEMFILES;
1   14:7   440                   EXIT(SPSYSTEMFILES);
1   14:6   444                   END;
1   14:5   444               IF(I<0) OR (I>10) THEN
1   14:6   457                 WRITELN('PLEASE TYPE AN INTEGER BETWEEN 0 AND 10')
1   14:4   516               UNTIL (I>=0)AND (I<=10);
1   14:4   529             OK:=FALSE;
1   14:4   533             IF I<>0 THEN
1   14:5   540               BEGIN
1   14:6   540                 OK:=TRUE;
```

SPSYSTEMFILES displays the name of the system files and determines which system the analyst wishes to use.

```
1   14:6   544                IF SPSYS[I]='' THEN
1   14:7   564                  BEGIN
1   14:8   564                    WRITELN(NSPSYS[I],' DOES NOT EXIST AT PRESENT');
1   14:8   633                    WRITELN('PLEASE TRY ANOTHER SYSTEM');
1   14:8   678                    OK:=FALSE
1   14:7   678                    END;
1   14:5   682                END
1   14:4   682              ELSE
1   14:5   684                BEGIN
1   14:6   684                  OK:=TRUE;
1   14:6   688                  OVER:=FALSE;
1   14:5   692                  END
1   14:3   692            UNTIL OK;
1   14:2   697        IF I=0 THEN
1   14:3   704          S5;
1   14:1   706      UNTIL OVER;
1   14:1   711    CURSP:=SPSYS[I];
1   14:1   729    NCURSP:=NSPSYS[I];
1   14:0   746    END;
1   14:0   770
```

See previous page for program description.

```
1 8:D   1(*$P*)PROCEDURE S5;
1 8:0   0  BEGIN
1 8:1   0    PREPKEY(57,'Would you like to add systems to this class of systems?');
1 8:1  61    IF (ANS='Y') OR (ANS='y') THEN
1 8:2  74      BEGIN
1 8:3  74        SPSYSCREATE;
1 8:3  76        OVER:=FALSE
1 8:2  76        END
1 8:1  80      ELSE
1 8:2  82        BEGIN
1 8:3  82          IF NDATA=0 THEN
1 8:4  89            S2;
1 8:3  91          PREPKEY(57,'Would you like to process another class of systems?');
1 8:3 148          IF ANS='Y' THEN
1 8:4 155            BEGIN
1 8:5 155              SYSTEMFILES;
1 8:5 157              SPSYSTEMFILES;
1 8:5 159              PASSNODE^.FLAG1:=0;
1 8:5 167              BRANCHOUT;
1 8:5 169              MENU;
1 8:4 171              END;
1 8:3 171          PREPKEY(2,'Would you like to stop for now?');
1 8:3 208          IF (ANS='Y') OR (ANS='y') THEN
1 8:4 221            QUIT
1 8:3 221            ELSE
1 8:4 225              BEGIN
1 8:5 225                WRITELN('THERE ARE NO MORE OPTIONS--SO I WILL PRESENT OPTIONS LIST AGAIN');
1 8:5 308                WRITELN('$$$Please press any Key to continue$$$');
1 8:5 366                (*$R-*)
1 8:5 366                READ(ANS);
1 8:5 377                (*$R+*)
1 8:4 377                END;
1 8:4 377
1 8:3 377          OVER:=FALSE;
1 8:2 381        END;
1 8:0 381    END;
1 8:0 398
1 8:0 398
1 8:0 398(*$I #5:GREET3.TEXT*)
1 8:0 398
```

S5 is a contuation of SPSYSTEMFILES.

```
1    1:0     0   BEGIN
1    1:1     0     BRANCHIN;
1    1:1   111     IF PASSNODE^.FLAG1<>1 THEN
1    1:2   121       BEGIN
1    1:3   121         HELLO;
1    1:3   123         PAGE(OUTPUT);
1    1:3   133         PREPKEY(1,'Would you like instructions (type yes or no, then press the
                       return key)?');
1    1:3   211         IF ANS='Y'  THEN
1    1:4   218           INSTRUCTIONS;
1    1:3   220         SYSTEMFILES;
1    1:3   222         SPSYSTEMFILES;
1    1:3   224         SUBSYSTEMFILES;
1    1:2   226         END;
1    1:1   226     BRANCHOUT;
1    1:1   228     MENU;
1    1:0   230     END.
```

MAINPROGRAM: If cold start 1) displays title page, 2) determines which system class/system/and subsystem analyst wants, 3) determines which analytic procedure analyst wants and 4) branches to appropriate analytic procedure for system class, system and subsystem selected.

PERFORMANCE ITEM PROGRAM (PERFITEM)

The performance item program allows the analyst to edit the performance items (objectives, functional purposes and characteristics), adding items, removing items, rewording items and printing out the items available.

```
 1    1:D     1 (*$L PRINTER:*)
 1    1:D     1 (*$S+*)
 1    1:D     1 PROGRAM Builddatafile;
 1    1:D     3 (*Program to process the performance items*)
 1    1:D     3 (*Ronald G. Shapiro     V2.0       10/25/82*)
 1    1:D     3
28    1:D     3
28    2:D     1   PROCEDURE SETCHAIN(TYTLE:STRING);
28    3:D     1   PROCEDURE SETCVAL(VAL:STRING);
28    4:D     1   PROCEDURE GETCVAL(VAR VAL:STRING);
28    5:D     1   PROCEDURE SWAPON;
28    6:D     1   PROCEDURE SWAPOFF;
28    6:D     1
 1    1:D     1 USES CHAINSTUFF;
 1    1:D     3
```

These procedures are part of the Apple Computer's CHAINSTUFF library entry. The demonstration package uses only SETCHAIN which causes another program to be activated.

```
16   1    1:D     3 (*$P*)CONST
17   1    1:D     3   OBJLBL1='The system must be capable of:';
18   1    1:D     3   OBJLBL2='The system must carry out the following activities:';
19   1    1:D     3   OBJLBL3='The system must produce:';
20   1    1:D     3   OBJLBL4='Performance objectives must be met despite:';
21   1    1:D     3   OBJLBL5='Performance objectives must be met despite:';
22   1    1:D     3
23   1    1:D     3   FPURLBL1='This system capability allows:';
24   1    1:D     3   FPURLBL2='The reasons for carrying out this activity are to:';
25   1    1:D     3   FPURLBL3='This product will be used by the system to:';
26   1    1:D     3   FPURLBL4='System purposes must be satisfied despite:';
27   1    1:D     3   FPURLBL5='System purposes must be satisfied despite:';
28   1    1:D     3
29   1    1:D     3   CHARLBL1='For this purpose, the system must have the potential for:';
30   1    1:D     3   CHARLBL2='The tasks required to satisfy this activity are to:';
31   1    1:D     3   CHARLBL3='To realize that product the system must first produce:';
32   1    1:D     3   CHARLBL4='Performance characteristics must be acceptable despite:';
33   1    1:D     3   CHARLBL5='Performance characteristics must be acceptable despite:';
34   1    1:D     3
```

Constants are defined.

```
 1    1:D     3 (*$P*)TYPE
 1    1:D     3
 1    1:D     3   PASSFILE=RECORD
 1    1:D     3     CURSYS,CURSP,CURSUB,PAC:STRING[80];
 1    1:D     3     NCURSYS,NCURSP,NCURSUB,NPAC,FLAG1,FLAG2,FLAG3:INTEGER;
 1    1:D     3     END;
 1    1:D     3
 1    1:D     3   DATABASE=RECORD
 1    1:D     3     NTAXA:ARRAY[1..4] OF INTEGER;
 1    1:D     3     TAXA:STRING[80];
 1    1:D     3     END;
 1    1:D     3
 1    1:D     3   HELPFILE=RECORD
 1    1:D     3     LINE:ARRAY[1..10] OF STRING[80];
 1    1:D     3     END;
 1    1:D     3
```

PASSFILE passes information about 1) system class [CURSYS, NCURSYS], 2) system [CURSP, NCURSP], 3) subsystem [CURSUB, NCURSUB], and 4) aspect [PAC, NPAC] from one program to another. Flag 1 is used to tell the GREETING program whether to begin with title page or analytic procedure list. Flags 2 and 3 are unused. DATABASE contains the performance items. HELPFILE contains the help commands.

```
51    1     1:D      3 (*$P*)VAR
52    1     1:D      3   XCHARAC,XFUNPUR,XOBJECTIVE,PAC,CURSYS,CURSP,CURSUB,ANSWER:STRING[80];
53    1     1:D    331   ANSHOLD,ANS2,ANS:CHAR;
54    1     1:D    334   DONE,OK,OVER,POS,NEG:BOOLEAN;
55    1     1:D    339   I,II,II2,J,K,L,M,N,NCHARAC,NFUNPUR,NOBJECTIVE,NPAC,NCURSYS,NCURSP,NCURSUB:INTEGER;
56    1     1:D    354   INLINECALL,PC,LLENGTH,NLENGTH,PGE,JHELP,TEMP2,LEAVE,HELP,NSCREEN,NPRINT,NDATA,
                         NCORELAST:INTEGER;
57    1     1:D    367   TSCR,TEMP,CORELAST,EII:INTEGER[8];
58    1     1:D    379   NAMEHELPFILE,NAMECOREFILE,NAMEDATAFILE:STRING[24];
59    1     1:D    418   REGLINE,LINE:STRING[80];
60    1     1:D    500   APMDSK:STRING[8];
61    1     1:D    505
62    1     1:D    505   ASPECT:ARRAY[1..5] OF STRING[14];
63    1     1:D    545
64    1     1:D    545   SCRATCH:ARRAY[1..20]OF STRING[80];
65    1     1:D   1365   NSCRATCH:ARRAY[1..20] OF INTEGER;
66    1     1:D   1385
67    1     1:D   1385   CORE:ARRAY[1..300] OF INTEGER[8];
68    1     1:D   2285
69    1     1:D   2285   COREFILE:FILE OF INTEGER[8];
70    1     1:D   2588   DATANODE:FILE OF DATABASE;
71    1     1:D   2933   PASSNODE:FILE OF PASSFILE;
72    1     1:D   3404   HELPER: FILE OF HELPFILE;
73    1     1:D   4114   PRNT:TEXT;
74    1     1:D   4415
```

These strings, arrays and variables are used by this program.

```
 75   1    2:D     1 (*$P*)PROCEDURE CORECLOSE;FORWARD;
 76   1    3:D     1 PROCEDURE CHARCREATE;FORWARD;
 77   1    4:D     1 PROCEDURE CHARACTERISTICS;FORWARD;
 78   1    5:D     1 PROCEDURE PCHARCREATE;FORWARD;
 79   1    6:D     1 PROCEDURE OBJECTIVES;FORWARD;
 80   1    7:D     1 PROCEDURE FCC;FORWARD;
 81   1    8:D     1 PROCEDURE INDEX;FORWARD;
 82   1    9:D     1 PROCEDURE DELFUN;FORWARD;
 83   1   10:D     1 PROCEDURE DELCAR;FORWARD;
 84   1   11:D     1 PROCEDURE DISPSCRATCH;FORWARD;
 85   1   12:D     1 PROCEDURE OBJCREATE;FORWARD;
 86   1   13:D     1 PROCEDURE HELPROUTINE;FORWARD;
 87   1   14:D     1 PROCEDURE OBJ7;FORWARD;
 88   1   15:D     1 PROCEDURE FPUR;FORWARD;
 89   1   15:D     1
 89   1   15:D     1   (*$I #5:PERFITEM2.TEXT *)
 90   1    1:D     1 VAR
 91   1    1:D  4415 INDENT:INTEGER;
 92   1    1:D  4416 LINEOK:BOOLEAN;
 93   1    1:D  4417 LONGLINE:STRING[125];
 94   1    1:D  4480
 95   1   16:D     1 PROCEDURE KEYN;FORWARD;
 96   1   17:D     1 PROCEDURE KEY;FORWARD;
 97   1   18:D     1 PROCEDURE BRANCHOUT;FORWARD;
 98   1   19:D     1 PROCEDURE REMOVE;FORWARD;
 99   1   20:D     1 PROCEDURE PREFIXO;FORWARD;
100   1   21:D     1 PROCEDURE PREFIXF;FORWARD;
101   1   22:D     1 PROCEDURE PREFIXC;FORWARD;
102   1   23:D     1 PROCEDURE ANYKEY;FORWARD;
103   1   23:D     1
```

These procedures are presented later on in this program.

```
1   24:D    1 (*$P*)PROCEDURE KEYNPREP(HLP:INTEGER;MSG:STRING);
1   24:0    0   BEGIN
1   24:1    0     HELP:=HLP;
1   24:1    9     WRITE(MSG);
1   24:1   20     KEYN;
1   24:0   22     END;
1   24:0   34
```

KEYNPREP displays a one line message, then calls KEYN to read a number from the keyboard.

```
1   25:D     1 (*$P*)PROCEDURE PREPKEY(HLP:INTEGER;MSG:STRING);
1   25:0     0   BEGIN
1   25:1     0     HELP:=HLP;
1   25:1     9     REPEAT
1   25:2     9       WRITE(MSG);
1   25:2    20       KEY;
1   25:1    22       UNTIL(ANS='Y') OR (ANS='N');
1   25:0    35     END;
1   25:0    50
```

PREPKEY displays a message then calls KEY to read a letter response from the keyboard. If a response is not Y, y, N, n, Yes or No, it redisplays the message and, once again, waits for a response.

```
1    26:D     1 (*$P*)PROCEDURE PROPERDISK;
1    26:0     0   BEGIN
1    26:1     0     REPEAT
1    26:1     0       (*$I-*)
1    26:2     0       RESET(HELPER,NAMEHELPFILE);
1    26:2    11       (*$I+*)
1    26:2    11       K:=IORESULT;
1    26:2    16       IF K=9 THEN
1    26:3    23         BEGIN
1    26:4    23           PAGE(OUTPUT);
1    26:4    33           WRITELN('Please reinsert your data disk into Drive # 2');
1    26:4    98           ANYKEY;
1    26:3   100           END;
1    26:1   100       UNTIL K<>9;
1    26:1   107       CLOSE(HELPER);
1    26:0   116       END;
1    26:0   130
```

PROPERDISK checks to be certain the appropriate subsystem's disk is in Drive #2.

```
1   27:D     1 (*$P*)PROCEDURE PRNTHELP;
1   27:0     0   BEGIN
1   27:1     0     DONE:=FALSE;
1   27:1     4     REWRITE(PRNT,'PRINTER:');
1   27:1    25     PAGE(PRNT);
1   27:1    35     WRITELN(PRNT,CHR(14),'Analytic Process Model',CHR(13));
1   27:1    97     WRITELN(PRNT,CHR(14),'Brief Help File',chr(13));
1   27:1   152     PGE:=2;
1   27:1   156     REPEAT
1   27:2   156       SEEK(HELPER,PGE);
1   27:2   167       GET(HELPER);
1   27:2   175       PAGE(PRNT);
1   27:2   185       K:=PGE-1;
1   27:2   193       WRITELN(PRNT,'                                                ',
1   27:2   245                 '                                 Page ',K);
1   27:2   314       FOR J:=1 TO 10 DO
1   27:3   328         WRITELN(PRNT,HELPER^.LINE[J]);
1   27:2   368       IF COPY(HELPER^.LINE[2],2,10)='conclusion' THEN
1   27:3   405         DONE:=TRUE;
1   27:2   409       PGE:=PGE+1;
1   27:1   417       UNTIL(DONE);
1   27:1   422     PAGE(PRNT);
1   27:1   432     CLOSE(PRNT);
1   27:0   441     END;
1   27:0   458
```

PRNTHELP prints the HELP file on the printer. It is called by HELPROUTINE.

```
1   13:D     1 (*$P*)PROCEDURE HELPROUTINE;
1   13:0     0   BEGIN
1   13:0     0     (*$I-*)
1   13:1     0     RESET(HELPER,'#5:HELP');
1   13:1    18     (*$I+*)
1   13:1    18     I:=IORESULT;
1   13:1    23     IF (I<>0) THEN
1   13:2    30       BEGIN
1   13:3    30         PAGE(OUTPUT);
1   13:3    40         WRITELN('UNFORTUNATELY, THE HELP FILE IS NOT AVAILABLE ON YOUR DISK');
1   13:3   118         WRITELN(' ');
1   13:3   136         WRITELN('PLEASE PRESS ANY KEY TO CONTINUE PROCESSING');
1   13:3   199         READ(ANS);
1   13:3   210         EXIT(HELPROUTINE);
1   13:2   214         END;
1   13:1   214       I:=0;
1   13:1   218       PGE:=HELP+1;
1   13:1   226       DONE:=FALSE;
1   13:1   230       REPEAT
1   13:2   230         SEEK(HELPER,PGE);
1   13:2   241         GET(HELPER);
1   13:2   249         PAGE(OUTPUT);
1   13:2   259         GOTOXY(73,0);
1   13:2   264         K:=PGE-1;
1   13:2   272         WRITELN('Page ',K);
1   13:2   309         GOTOXY(0,0);
1   13:2   314         FOR J:=1 TO 10 DO
1   13:3   328           WRITELN(HELPER^.LINE[J]);
1   13:2   368         IF COPY(HELPER^.LINE[2],2,10)='conclusion' THEN
1   13:3   405           DONE:=TRUE;
1   13:2   409         WRITELN(' ');
1   13:2   427         WRITELN('****PLEASE PRESS RETURN KEY TO VIEW NEXT PAGE****');
1   13:2   496         WRITELN('****PLEASE TYPE PAGE NUMBER AND PRESS RETURN KEY TO VIEW
                       ANOTHER PAGE****');
1   13:2   589         WRITE  ('****PLEASE PRESS ESC AND RETURN KEYS TO ESCAPE HELP ROUTINE****');
1   13:2   664         PGE:=PGE+1;
1   13:2   672         (*$R-*)
1   13:2   672         ANSWER:='          ';
1   13:2   689         READLN(ANSWER);
1   13:2   708         page(output);
```

HELPROUTINE displays appropriate help commands when it is called by KEY or KEYN. HELPROUTINE knows which HELP to display because the calling program places the appropriate help page number into HELP. Once the analyst sees the first help message, he/she can ask for other help messages by typing in the page number of the desired help messages. Note that the HELP file is made by editing a series of files (HELP1 . . . HELPN) using the Apple editor. Then, they are processed by the BLOCKHELP program (see Chapter VIII). The HELP file produced by BLOCKHELP is suitable for use with the HELPROUTINE. HELPROUTINE "knows" it has hit the last page of the file because the word "conclusion" appears on the second line of the last page.

```
1   13:2   718        IF ORD(ANSWER[1])=27 THEN
1   13:3   726          BEGIN
1   13:4   726            CLOSE(HELPER);
1   13:4   735            PROPERDISK;
1   13:4   737            (*$R+*)
1   13:4   737            EXIT (HELPROUTINE);
1   13:4   741            (*$R-*)
1   13:3   741            END;
1   13:2   741        IF (ANSWER[1]>='0') AND (ANSWER[1]<='9') THEN
1   13:3   756          BEGIN
1   13:4   756            PGE:=ORD(ANSWER[1])-48;
1   13:4   765            IF (ANSWER[2]>='0') AND (ANSWER[2]<='9') THEN
1   13:5   780              PGE:=PGE*10 + ORD(ANSWER[2])-48;
1   13:4   795            PGE:=PGE+1;
1   13:4   803            IF PGE<2 THEN
1   13:5   810              PGE:=2;
1   13:4   814            DONE:=FALSE;
1   13:3   818            END;
1   13:1   818        UNTIL (DONE) AND ((ANSWER[1]<'0') OR (ANSWER[1]>'9'));
1   13:1   837    PAGE(OUTPUT);
1   13:1   847    PREPKEY(2,'Would you like to print the help file?');
1   13:1   891    IF ANS='Y' THEN
1   13:2   898      BEGIN
1   13:3   898        KEYNPREP(2,'How many copies? ');
1   13:3   921        FOR N:=1 TO I DO
1   13:4   937          PRNTHELP;
1   13:3   949        WRITELN('DONE');
1   13:2   973        END;
1   13:1   973    CLOSE(HELPER);
1   13:1   982    PROPERDISK;
1   13:1   984    (*$R+*)
1   13:0   984    END;
1   13:0  1004
```

See previous page for program description.

```
1   17:D     1 (*$P*)PROCEDURE KEY;
1   17:0     0   BEGIN
1   17:0     0     (*$R-*)
1   17:1     0     ANSWER:='                 ';
1   17:1    24     REPEAT
1   17:2    24       READLN(ANSWER);
1   17:2    43       ANS:=ANSWER[1];
1   17:2    50       IF (ANS<>'Y')AND(ANS<>'N')AND(ANS<>'H')AND(ANS<>'y')AND
1   17:2    73         (ANS<>'n')AND(ANS<>'h') THEN
1   17:3    87         WRITELN('PLEASE RESPOND YES OR NO!');
1   17:2   132       IF ORD(ANS)>90 THEN
1   17:3   139         BEGIN
1   17:4   139           II2:=ORD(ANS)-32;
1   17:4   147           ANS:=CHR(II2);
1   17:3   153           END;
1   17:1   153       UNTIL (ANS='Y') OR (ANS='N') OR (ANS='H');
1   17:1   172       (*$R+*)
1   17:1   172       IF ANS='H' THEN
1   17:2   179         HELPROUTINE;
1   17:0   181       END;
1   17:0   196
```

KEY reads a letter response from the keyboard. If response is 1) y or Y, it places a Y in ANS and returns to calling procedure; 2) n or N, it places an N in ANS and returns to calling procedure; 3) h or H, it calls the HELP routine, places an H in ANS and returns to calling program; or 4) any other key—it displays PLEASE RESPOND YES OR NO and awaits a Y, N, H, y, n or h response. NOTE: Only the first character/line is processed. The rest is ignored.

```
1   16:D     1 (*$P*)PROCEDURE KEYN;
1   16:0     0   BEGIN
1   16:0     0     (*$R-*)
1   16:1     0     ANSWER:='                    ';
1   16:1    25     REPEAT
1   16:2    25       REPEAT
1   16:3    25         READLN(ANSWER);
1   16:3    44         IF LENGTH(ANSWER)=0 THEN
1   16:4    52           WRITELN('Please enter the integer again');
1   16:2   102         UNTIL LENGTH(ANSWER)<>0;
1   16:2   110       ANS:=ANSWER[1];
1   16:2   117       ANS2:=ANSWER[2];
1   16:2   124       IF (ANS='H') OR (ANS='h') THEN
1   16:3   137         BEGIN
1   16:4   137           HELPROUTINE;
1   16:4   139           I:=999;
1   16:4   145           EXIT(KEYN);
1   16:3   149           END;
1   16:2   149       II:=ORD(ANS)-48;
1   16:2   157       II2:=-1;
1   16:2   162       II2:=ORD(ANS2)-48;
1   16:2   170       IF (II<0) OR (II>9) THEN
1   16:3   183         WRITELN('PLEASE RESPOND WITH AN INTEGER!');
1   16:1   234       UNTIL (II>=0)AND (II<10);
1   16:1   247       I:=II;
1   16:1   253       IF (II2>=0)AND(II2<=9) THEN
1   16:2   266         I:=II*10+II2;
1   16:2   278       (*$R+*)
1   16:0   278       END;
1   16:0   294
```

KEYN reads a 1 or 2 digit response from the keyboard and places it into I. If an H or an h are typed in, it places a 999 in I and calls the HELP routine. If more than 2 characters are typed, only 2 characters are read. The rest are ignored. If the character(s) are not positive intergers, KEYN will display an appropriate warning and wait for a response.

```
1   23:D     1 {$$P$}PROCEDURE ANYKEY;
1   23:0     0   BEGIN
1   23:1     0     WRITELN(' ');
1   23:1    18     WRITELN('$$$ Please press any key to continue $$$');
1   23:1    78     {$$R-$}
1   23:1    78     READ(ANS);
1   23:1    89     {$$R+$}
1   23:0    89     END;
1   23:0   102
```

ANYKEY displays "Please Press any Key to Continue" then it awaits a Keypress before returning control to the calling procedure.

```
1   18:D     1   (*$P*)PROCEDURE BRANCHOUT;
1   18:0     0     BEGIN
1   18:0     0       (*$I-*)
1   18:1     0       REWRITE(PASSNODE,'PASSTHRU');
1   18:1    19       (*$I+*)
1   18:1    19       IF(IORESULT<>0) THEN
1   18:2    25         WRITE('SERIOUS ERROR -- NO FILE PASSTHRU AT BRANCHOUT')
1   18:1    83       ELSE
1   18:2    85         BEGIN
1   18:3    85           PASSNODE^.CURSYS:=CURSYS;
1   18:3    94           PASSNODE^.CURSP:=CURSP;
1   18:3   103           PASSNODE^.CURSUB:=CURSUB;
1   18:3   112           PASSNODE^.PAC:=PAC;
1   18:3   120           PASSNODE^.NCURSYS:=NCURSYS;
1   18:3   130           PASSNODE^.NCURSP:=NCURSP;
1   18:3   140           PASSNODE^.NCURSUB:=NCURSUB;
1   18:3   150           PASSNODE^.NPAC:=NPAC;
1   18:3   160           PASSNODE^.FLAG1:=1;
1   18:3   168           PUT(PASSNODE);
1   18:3   176           IF EOF(PASSNODE) THEN
1   18:4   186             WRITELN('OUT OF DISK SPACE WHILE WRITING PASSTHRU');
1   18:3   246           CLOSE(PASSNODE,LOCK);
1   18:2   255           END;
1   18:0   255       END;
1   18:0   270
```

BRANCHOUT loads the PASSTHRU file with appropriate data for use by called programs.

```
1   28:D     1   (*$P*)PROCEDURE BRANCHIN;
1   28:0     0     BEGIN
1   28:0     0       (*$I-*)
1   28:1     0       RESET(PASSNODE,'PASSTHRU');
1   28:1    19       (*$I+*)
1   28:1    19       IF IORESULT<>0 THEN
1   28:2    25         BEGIN
1   28:3    25           REWRITE(PASSNODE,'PASSTHRU');
1   28:3    46           PASSNODE^.CURSYS:='';
1   28:3    56           PASSNODE^.CURSP:='';
1   28:3    66           PASSNODE^.CURSUB:='';
1   28:3    76           PASSNODE^.PAC:='';
1   28:3    84           PASSNODE^.NCURSYS:=0;
1   28:3    92           PASSNODE^.NCURSP:=0;
1   28:3   100           PASSNODE^.NCURSUB:=0;
1   28:3   108           PASSNODE^.NPAC:=0;
1   28:3   116           PUT(PASSNODE);
1   28:3   124           IF EOF(PASSNODE) THEN
1   28:4   134             WRITELN('OUT OF DISK SPACE WHILE WRITING PASSTHRU');
1   28:3   194           CLOSE(PASSNODE,LOCK);
1   28:3   203           RESET(PASSNODE,'PASSTHRU')
1   28:2   224           END;
1   28:1   224       GET(PASSNODE);
1   28:1   232       CURSYS:=PASSNODE^.CURSYS;
1   28:1   241       CURSP:=PASSNODE^.CURSP;
1   28:1   250       CURSUB:=PASSNODE^.CURSUB;
1   28:1   259       PAC:=PASSNODE^.PAC;
1   28:1   267       NCURSYS:=PASSNODE^.NCURSYS;
1   28:1   276       NCURSP:=PASSNODE^.NCURSP;
1   28:1   285       NCURSUB:=PASSNODE^.NCURSUB;
1   28:1   294       NPAC:=PASSNODE^.NPAC;
1   28:1   303       CLOSE(PASSNODE);
1   28:0   312       END;
1   28:0   326
```

BRANCHIN gets information from the PASSTHRU file for use by this program.

```
1   29:D    1   (*$P*)PROCEDURE INLINE;
1   29:0    0     BEGIN
1   29:1    0       REPEAT
1   29:2    0         READLN(LONGLINE);
1   29:2   20         LINEOK:=TRUE;
1   29:2   24         M:=LENGTH(LONGLINE);
1   29:2   32         IF M>80 THEN
1   29:3   39           BEGIN
1   29:4   39             WRITELN('**WARNING LINE CONTAINS OVER 80 CHARACTERS**');
1   29:4  103             WRITELN(' ');
1   29:4  121             WRITELN('DO YOU WISH TO TRUNCATE TO 80 CHARACTERS?');
1   29:4  182             REPEAT
1   29:5  182               HELP:=39;
1   29:5  186               KEY
1   29:4  186               UNTIL (ANS='Y') OR (ANS='N');
1   29:4  201             IF ANS='N' THEN
1   29:5  208               BEGIN
1   29:6  208                 LINEOK:=FALSE;
1   29:6  212                 WRITELN('PLEASE RE-ENTER LINE:');
1   29:5  253                 END
1   29:4  253               ELSE
1   29:5  255                 M:=80;
1   29:3  259             END;
1   29:1  259         UNTIL LINEOK;
1   29:1  264       INLINECALL:=INLINECALL+1;
1   29:1  272       IF INLINECALL>25 THEN
1   29:2  279         BEGIN
1   29:3  279           WRITELN('WARNING: You have entered over 25 new performance items',
1   29:3  346           chr(13),'  during this session.  This is the limit allowed in the',
1   29:3  424           chr(13),'  demonstration system.  To enter more, please Select a',
1   29:3  501           chr(13),'  Different Analytic Procedure.  This will re-initialize',
1   29:3  579           chr(13),'  the stack pointer and allow you to enter more items!');
1   29:3  663           ANYKEY;
1   29:2  665           END;
1   29:1  665       SCRATCH[I]:=COPY(LONGLINE,1,M);
1   29:0  694       END;
1   29:0  714
```

INLINE accepts up to 80 characters of text. If more than 80 characters are specified, it asks if it ought to ignore additional characters. If told to, it does. Otherwise, it allows analyst to re-enter the line.

```
  1   30:D     1 (*$P*)PROCEDURE SHOWALINE;
  1   30:0     0   BEGIN
  1   30:1     0     NLENGTH:=LENGTH(LINE);
  1   30:1     8     IF NLENGTH<2 THEN
  1   30:2    15       EXIT(SHOWALINE);
  1   30:1    19     WHILE (LINE[NLENGTH]=' ') AND (NLENGTH>1) DO
  1   30:2    37       NLENGTH:=NLENGTH-1;
  1   30:1    47     IF NLENGTH<2 THEN
  1   30:2    54       EXIT(SHOWALINE);
  1   30:1    58     IF NLENGTH<=LLENGTH THEN
  1   30:2    67       BEGIN
  1   30:3    67         WRITE(LINE);
  1   30:3    79         EXIT(SHOWALINE);
  1   30:2    83         END;
  1   30:1    83     L:=LLENGTH;
  1   30:1    89     WHILE (LINE[L]<>' ') AND (L>1) DO
  1   30:2   107       L:=L-1;
  1   30:1   117     L:=L-1;
  1   30:1   125     IF L>1 THEN
  1   30:2   132       BEGIN
  1   30:3   132       REGLINE:=COPY(LINE,1,L);
  1   30:3   151       WRITELN(REGLINE);
  1   30:2   171       END;
  1   30:1   171     L:=L+2;
  1   30:1   179     NLENGTH:=NLENGTH-L+1;
  1   30:1   191     IF NLENGTH<1 THEN
  1   30:2   198       EXIT(SHOWALINE);
  1   30:1   202     REGLINE:=COPY(LINE,L,NLENGTH);
  1   30:1   223     WRITE('      ',REGLINE);
  1   30:1   253     PC:=PC+1;
  1   30:0   261     END;
  1   30:0   278
```

SHOWALINE displays text on the screen. If, by chance, the text is longer than the amount of space available on the current line, the display continues onto a second line.

```
1   31:D     1 (*$P*)PROCEDURE PRINTTOP;
1   31:0     0   BEGIN
1   31:1     0     M:=LENGTH(CURSYS);
1   31:1     7     IF M>16 THEN
1   31:2    14       M:=16;
1   31:1    18     LINE:=COPY(CURSYS,1,M);
1   31:1    36     WRITE(PRNT,'*',LINE,' Systems');
1   31:1    78     N:=16-LENGTH(CURSYS);
1   31:1    87     FOR L:=1 TO N DO
1   31:2   103       WRITE(PRNT,' ');
1   31:1   123     M:=LENGTH(CURSP);
1   31:1   130     IF M>16 THEN
1   31:2   137       M:=16;
1   31:1   141     LINE:=COPY(CURSP,1,M);
1   31:1   159     WRITE(PRNT,'*',LINE);
1   31:1   181     N:=16-LENGTH(CURSP);
1   31:1   190     FOR L:=1 TO N DO
1   31:2   206       WRITE(PRNT,' ');
1   31:1   226     M:=LENGTH(CURSUB);
1   31:1   233     IF M>16 THEN
1   31:2   240       M:=16;
1   31:1   244     LINE:=COPY(CURSUB,1,M);
1   31:1   262     WRITE(PRNT,'*',LINE);
1   31:1   284     N:=16-LENGTH(CURSUB);
1   31:1   293     FOR L:=1 TO N DO
1   31:2   309       WRITE(PRNT,' ');
1   31:1   329     WRITELN(PRNT,'*',PAC);
1   31:1   359     IF NPRINT>1 THEN
1   31:2   366       WRITELN(PRNT,'Objective:[',NOBJECTIVE,']',XOBJECTIVE);
1   31:1   431     IF NPRINT>2 THEN
1   31:2   438       WRITELN(PRNT,'Fctl Prps:[',NFUNPUR,']',XFUNPUR);
1   31:1   503     WRITELN(PRNT,' ');
1   31:0   521     END;
1   31:0   540
```

PRINT-TOP prints the current system class, system, subsystem, etc., on the printer.

```
 1   32:D     1 (*$P*)PROCEDURE PRINTSCRN;
 1   32:0     0   BEGIN
 1   32:1     0     REWRITE(PRNT,'PRINTER:');
 1   32:1    21     PAGE(PRNT);
 1   32:1    31     PRINTTOP;
 1   32:1    33     CLOSE(PRNT);
 1   32:1    42     CLOSE(OUTPUT);
 1   32:1    51     REWRITE(OUTPUT,'PRINTER:');
 1   32:1    72     IF NPRINT=1 THEN
 1   32:2    79       BEGIN
 1   32:3    79         WRITE('Objectives--');
 1   32:3   103         PREFIXO;
 1   32:2   105         END;
 1   32:1   105     IF NPRINT=2 THEN
 1   32:2   112       BEGIN
 1   32:3   112         WRITE('Functional purposes--');
 1   32:3   145         PREFIXF;
 1   32:2   147         END;
 1   32:1   147     IF NPRINT=3 THEN
 1   32:2   154       BEGIN
 1   32:3   154         WRITE('Characteristics--');
 1   32:3   183         PREFIXC;
 1   32:2   185         END;
 1   32:1   185     CLOSE(OUTPUT);
 1   32:1   194     REWRITE(OUTPUT,'CONSOLE:');
 1   32:1   215     REWRITE(PRNT,'PRINTER:');
 1   32:1   236     WRITELN(PRNT,' ');
 1   32:1   254     FOR K:=1 TO 20 DO
 1   32:2   268       BEGIN
 1   32:3   268         IF SCRATCH[K]<>'' THEN
 1   32:4   288           BEGIN
 1   32:5   288             NDATA:=1;
 1   32:5   292             WRITELN(PRNT,' ',NSCRATCH[K],'.  ',SCRATCH[K])
 1   32:4   370             END;
 1   32:2   370         END;
 1   32:1   380    IF NDATA=0 THEN
 1   32:2   387      WRITELN(PRNT,'I have no data at this time!!!!!');
 1   32:1   439    CLOSE(PRNT);
 1   32:0   448    END;
 1   32:0   462
```

PRINTSCRN prints the performance items currently being displayed on the screen.

```
1   33:D    1  (*$P*)PROCEDURE TOPSCREEN;
1   33:0    0    BEGIN
1   33:1    0      PAGE(OUTPUT);
1   33:1   10      M:=LENGTH(CURSYS);
1   33:1   17      IF M>16 THEN
1   33:2   24        M:=16;
1   33:1   28      LINE:=COPY(CURSYS,1,M);
1   33:1   46      WRITE('*',LINE,' Systems');
1   33:1   88      GOTOXY(26,0);
1   33:1   93      M:=LENGTH(CURSP);
1   33:1  100      IF M>16 THEN
1   33:2  107        M:=16;
1   33:1  111      LINE:=COPY(CURSP,1,M);
1   33:1  129      WRITE('*',LINE);
1   33:1  151      GOTOXY(44,0);
1   33:1  156      M:=LENGTH(CURSUB);
1   33:1  163      IF M>16 THEN
1   33:2  170        M:=16;
1   33:1  174      LINE:=COPY(CURSUB,1,M);
1   33:1  192      WRITELN('*',LINE);
1   33:1  222      GOTOXY(62,0);
1   33:1  227      WRITELN('*',PAC);
1   33:1  257      M:=LENGTH(XOBJECTIVE);
1   33:1  265      IF M>67 THEN M:=67;
1   33:1  276      LINE:=COPY(XOBJECTIVE,1,M);
1   33:1  295      IF NSCREEN>1 THEN
1   33:2  302        WRITELN('Objective[',NOBJECTIVE,']:',LINE);
1   33:1  370      M:=LENGTH(XFUNPUR);
1   33:1  378      IF M>67 THEN M:=67;
1   33:1  389      LINE:=COPY(XFUNPUR,1,M);
1   33:1  408      IF NSCREEN>2 THEN
1   33:2  415        WRITELN('Fctl Prps[',NFUNPUR,']:',LINE);
1   33:1  483      WRITELN(' ');
1   33:0  501      END;
1   33:0  514
```

TOPSCREEN displays the system class, system, subsystem, etc., on the top of the display screen.

```
1    34:D     1 (*$P*)PROCEDURE OPENCOREFILE;
1    34:0     0   BEGIN
1    34:0     0   (*$I-*)
1    34:1     0   RESET(COREFILE,NAMECOREFILE);
1    34:1    11   (*$I+*)
1    34:1    11   I:=IORESULT;
1    34:1    16   IF I<>0 THEN
1    34:2    23     BEGIN
1    34:3    23       REWRITE(COREFILE,NAMECOREFILE);
1    34:3    36       FOR I:=1 TO 300 DO
1    34:4    52         BEGIN
1    34:5    52           CORE[I]:=0;
1    34:5    79           COREFILE^:=CORE[I];
1    34:5   107           PUT(COREFILE);
1    34:5   115           IF EOF(COREFILE) THEN
1    34:6   125             BEGIN
1    34:7   125               WRITELN('OUT OF DISK SPACE!!!');
1    34:7   165               ANYKEY;
1    34:7   167               BRANCHOUT;
1    34:7   169               SETCHAIN('GREETING');
1    34:7   183               EXIT(PROGRAM);
1    34:6   187               END;
1    34:4   187           END;
1    34:3   197         CORELAST:=0;
1    34:3   212         NCORELAST:=0;
1    34:3   216         COREFILE^:=CORELAST;
1    34:3   232         PUT(COREFILE);
1    34:3   240         CLOSE(COREFILE,LOCK)
1    34:2   249         END
1    34:1   249     ELSE
1    34:2   251       BEGIN
1    34:3   251         FOR I:=1 TO 300 DO
1    34:4   267           BEGIN
1    34:5   267             GET(COREFILE);
1    34:5   275             CORE[I]:=COREFILE^;
1    34:4   303             END;
1    34:3   313         GET(COREFILE);
1    34:3   321         CORELAST:=COREFILE^;
1    34:3   337         NCORELAST:=TRUNC(CORELAST);
1    34:3   350         CLOSE (COREFILE)
1    34:2   359         END;
1    34:0   359     END;
1    34:0   380
```

OPENCOREFILE reads the index to the performance item file into core.

```
1   35:D     1 (*$P*)PROCEDURE OPENOBJFILE;
1   35:0     0   BEGIN
1   35:0     0   (*$I-*)
1   35:1     0   RESET(DATANODE,NAMEDATAFILE);
1   35:1    11   (*$I+*)
1   35:1    11   IF IORESULT<>0 THEN
1   35:2    17     BEGIN
1   35:3    17       WRITELN('Please bear with me while I make room for your ',
1   35:3    76               'analysis on the disk');
1   35:3   116       REWRITE(DATANODE,NAMEDATAFILE);
1   35:3   129       FOR I:=1 TO 4 DO
1   35:4   143         DATANODE^.NTAXA[I]:=0;
1   35:3   168       FOR I:=1 TO 300 DO
1   35:4   184         BEGIN
1   35:5   184           DATANODE^.TAXA:='';
1   35:5   194           SEEK(DATANODE,I);
1   35:5   205           PUT(DATANODE);
1   35:5   213           IF EOF(DATANODE) THEN
1   35:6   223             BEGIN
1   35:7   223               WRITELN('OUT OF DISK SPACE!!!')
1   35:6   263               END;
1   35:4   263           END;
1   35:3   273         CLOSE(DATANODE,LOCK);
1   35:3   282         RESET(DATANODE,NAMEDATAFILE)
1   35:2   295         END;
1   35:0   295     END;
1   35:0   314
1   35:0   314
1   35:0   314   (*$I #5:PERFITEM2.TEXT *)
1   35:0   314
1   35:0   314
```

OPENOBJFILE creates the performance item file if it does not already exist.

```
1   22:D     1 (*$P*)PROCEDURE PREFIXC;
1   22:0     0   BEGIN
1   22:1     0     CASE NPAC OF
1   22:1     5       1: WRITELN(CHARLBL1);
1   22:1    84       2: WRITELN(CHARLBL2);
1   22:1   157       3: WRITELN(CHARLBL3);
1   22:1   233       4: WRITELN(CHARLBL4);
1   22:1   310       5: WRITELN(CHARLBL5);
1   22:1   387       END;
1   22:0   404     END;
1   22:0   420
```

PREFIXC displays sentence prefixes for a characteristic.

```
1    21:D      1 (*$P*)PROCEDURE PREFIXF;
1    21:0      0   BEGIN
1    21:1      0     CASE NPAC OF
1    21:1      5       1: WRITELN(FPURLBL1);
1    21:1     57       2: WRITELN(FPURLBL2);
1    21:1    129       3: WRITELN(FPURLBL3);
1    21:1    194       4: WRITELN(FPURLBL4);
1    21:1    258       5: WRITELN(FPURLBL5);
1    21:1    322       END;
1    21:0    340     END;
1    21:0    356
```

PREFIXF displays sentence prefixes for a functional purpose.

```
  1   20:D     1 (*$P*)PROCEDURE PREFIXO;
  1   20:0     0   BEGIN
  1   20:1     0     CASE NPAC OF
  1   20:1     5       1: WRITELN(OBJLBL1);
  1   20:1    57       2: WRITELN(OBJLBL2);
  1   20:1   130       3: WRITELN(OBJLBL3);
  1   20:1   176       4: WRITELN(OBJLBL4);
  1   20:1   241       5: WRITELN(OBJLBL5);
  1   20:1   306       END;
  1   20:0   324     END;
  1   20:0   340
```

PREFIXO displays sentence prefixes for an objective.

```
 1   36:D     1 (*$P*)PROCEDURE ASPECTS;
 1   36:0     0   BEGIN
 1   36:1     0     REPEAT
 1   36:2     0       PAGE(OUTPUT);
 1   36:2    10       WRITELN('You are currently analyzing subsystem ',CURSUB,CHR(13),'  of the ',
 1   36:2   102               CURSYS,' class of systems');
 1   36:2   150       WRITELN(' ');
 1   36:2   168       ASPECT[1]:='Potentialities';
 1   36:2   198       ASPECT[2]:='Processes';
 1   36:2   223       ASPECT[3]:='Products';
 1   36:2   247       ASPECT[4]:='Environment';
 1   36:2   274       ASPECT[5]:='Constraints';
 1   36:2   301       HELP:=8;
 1   36:2   305       WRITELN('To proceed with the analysis you may examine the following');
 1   36:2   383       WRITELN('  aspects of performance:');
 1   36:2   428       FOR I:=1 TO 5 DO
 1   36:3   442         WRITELN('  ',I,'.  ',ASPECT[I]);
 1   36:2   523       WRITELN('  0.  Select a different analytic procedure');
 1   36:2   586       WRITELN(' ');
 1   36:2   604       WRITE('Which aspect of subsystem ',CURSUB,' would you like to analyze?');
 1   36:2   692       REPEAT
 1   36:3   692         KEYN;
 1   36:3   694         IF((I<0) OR (I>5)) AND (I<>999) THEN
 1   36:4   715           WRITELN('PLEASE SELECT AN INTEGER BETWEEN 0 AND 5');
 1   36:2   775         UNTIL((I>=0) AND (I<=5)) OR (I=999);
 1   36:1   796       UNTIL I<>999;
 1   36:1   805     IF I=0 THEN
 1   36:2   812       BEGIN
 1   36:3   812         CORECLOSE;
 1   36:3   814         BRANCHOUT;
 1   36:3   816         SETCHAIN('GREETING');
 1   36:3   830         EXIT (PROGRAM);
 1   36:2   834         END;
 1   36:1   834     PAC:=ASPECT[I];
 1   36:1   852     NPAC:=I;
 1   36:1   858     WRITELN('You have chosen to analyze the ',PAC,' aspect',CHR(13),' of subsystem ',
 1   36:1   969             CURSUB,' performance.');
 1   36:0  1013     END;
 1   36:0  1032
```

ASPECTS allows analyst to select the aspect he/she intends to use.

```
1   37:D     1  (*$P*)PROCEDURE FPUR1;
1   37:0     0    BEGIN
1   37:1     0      GOTOXY(0,16);
1   37:1     5      WRITE(CHR(11));
1   37:1    15        REPEAT
1   37:2    15          WRITE('Which functional purpose would you like to analyze (Type 0 to
                        reconsider)?');
1   37:2   101          HELP:=15;
1   37:2   105          KEYN;
1   37:2   107          IF I=999 THEN
1   37:3   116            FPUR;
1   37:2   118          IF(I<0)OR(I>20)THEN
1   37:3   131            WRITELN('PLEASE INPUT AN INTEGER BETWEEN 1 AND 20');
1   37:2   191          IF(I>0)AND(I<20)THEN
1   37:3   204            IF SCRATCH[I]='' THEN
1   37:4   224              BEGIN
1   37:5   224                WRITELN(I,' DOES NOT EXIST AT PRESENT');
1   37:5   282                WRITELN('PLEASE TRY ANOTHER FUNCTIONAL PURPOSE');
1   37:5   339                I:=25;
1   37:4   343                END;
1   37:1   343          UNTIL (I>=0) AND (I<20);
1   37:1   356            IF I>0 THEN
1   37:2   363              BEGIN
1   37:3   363                NFUNPUR:=NSCRATCH[I];
1   37:3   380                XFUNPUR:=SCRATCH[I];
1   37:3   398                CHARACTERISTICS;
1   37:2   400                END;
1   37:0   400      END;
1   37:0   416
```

FPUR1 asks the analyst which functional purpose he/she would like to analyze when he/she requests to analyze characteristics.

```
1   38:D     1 (*$P*)PROCEDURE FUNCCREATE;
1   38:0     0   BEGIN
1   38:1     0     REPEAT
1   38:2     0       NSCREEN:=2;
1   38:2     4       TOPSCREEN;
1   38:2     6       WRITELN('You have chosen to create a new functional purpose.');
1   38:2    77       WRITELN(' ');
1   38:2    95       INDEX;
1   38:2    97       IF I=0 THEN
1   38:3   104         EXIT(FUNCCREATE);
1   38:2   108       FCC;
1   38:1   110       UNTIL OK;
1   38:0   115     END;
1   38:0   130
```

FUNCCREATE helps analysts to create new functional purposes by finding an appropriate index for the new purpose.

```
719   1    7:D     1  (*$P*)PROCEDURE FCC;
720   1    7:0     0    BEGIN
721   1    7:1     0      WRITELN('Please specify (80 additional characters available) the new ',
722   1    7:1    72              'functional purpose ',chr(13),' within the ',PAC,
723   1    7:1   149              ' aspect of the ',CURSUB,' system.');
724   1    7:1   215      WRITE(CHR(13));
725   1    7:1   225      PREFIXF;
726   1    7:1   227      INLINE;
727   1    7:1   229      IF SCRATCH[I]='' THEN
728   1    7:2   249        EXIT(FUNCCREATE);
729   1    7:1   253      NSCRATCH[I]:=I;
730   1    7:1   270      WRITELN('Done');
731   1    7:1   294      DATANODE^.NTAXA[1]:=NPAC;
732   1    7:1   309      DATANODE^.NTAXA[2]:=NOBJECTIVE;
733   1    7:1   324      DATANODE^.NTAXA[3]:=NSCRATCH[I];
734   1    7:1   350      DATANODE^.NTAXA[4]:=0;
735   1    7:1   363      DATANODE^.TAXA:=SCRATCH[I];
736   1    7:1   383      REPEAT
737   1    7:2   383        BEGIN
738   1    7:3   383          J:=TRUNC(CORELAST);
739   1    7:3   396          IF J>=300 THEN
740   1    7:4   405            BEGIN
741   1    7:5   405              WRITELN('*** ERROR -- YOUR DATA SET ALREADY CONTAINS 300 PERFORMANCE
                                                 ITEMS! ***');
742   1    7:5   495              WRITELN('***          THUS, THIS ITEM WAS NOT ADDED TO DATA SET ***');
743   1    7:5   573              ANYKEY;
744   1    7:5   575              EXIT(FUNCCREATE);
745   1    7:4   579              END;
746   1    7:3   579          CORELAST:=CORELAST+1;
747   1    7:3   604          J:=J+1;
748   1    7:3   612          NCORELAST:=J;
749   1    7:3   618          EII:=CORE[J] DIV 1000000;
750   1    7:2   675          END;
751   1    7:1   675        UNTIL EII =0;
752   1    7:1   693      TEMP:=NPAC;
753   1    7:1   710      TSCR:=NOBJECTIVE;
754   1    7:1   727      CORE[J]:=TEMP*1000000+TSCR*10000+NSCRATCH[I]*100+0;
755   1    7:1   838      SEEK(DATANODE,J);
756   1    7:1   849      PUT(DATANODE);
757   1    7:1   857      WRITELN('It will be necessary, at some time, to add characteristics to this',
                          chr(13),'function
```

FCC accepts the new functional purpose and stores it in the performance item data set.

```
758 1 7:1  987      PREPKEY(17,'Would you like to specify characteristics at this time?');
759 1 7:1 1048      IF ANS='Y' THEN
760 1 7:2 1055        BEGIN
761 1 7:3 1055          XFUNPUR:=SCRATCH[I];
762 1 7:3 1073          NFUNPUR:=NSCRATCH[I];
763 1 7:3 1090          PCHARCREATE;
764 1 7:3 1092          CHARCREATE;
765 1 7:3 1094          EXIT(FUNCCREATE);
766 1 7:2 1098          END;
767 1 7:1 1098      PREPKEY(15,'Would you like to specify more functional purposes for this objective?');
768 1 7:1 1174      IF ANS='Y' THEN
769 1 7:2 1181        OK:=FALSE
770 1 7:1 1181        ELSE
771 1 7:2 1187          EXIT(FUNCCREATE);
772 1 7:0 1191      END;
773 1 7:0 1208
```

See previous page for program description.

```
1    9:D     1 (*$P*)PROCEDURE DELFUN;
1    9:0     0   BEGIN
1    9:1     0     REPEAT
1    9:2     0       GOTOXY(0,16);
1    9:2     5       WRITE(chr(11),'Which one do you want to remove(Type 0 to reconsider):');
1    9:2    81       HELP:=15;
1    9:2    85       KEYN;
1    9:2    87       IF I=999 THEN
1    9:3    96         FPUR;
1    9:2    98       IF (I<0) OR (I>20) THEN
1    9:3   111         WRITELN('PLEASE TYPE AN INTEGER BETWEEN 0 AND 20');
1    9:1   170       UNTIL (I>=0)AND(I<21);
1    9:1   183     IF I=0 THEN
1    9:2   190       BEGIN
1    9:3   190         EXIT(DELFUN);
1    9:2   194         END;
1    9:1   194     PREPKEY(15,'Do you really want to remove this functional purpose & assoc
               characteristics ?');
1    9:1   277     IF ANS='N' THEN
1    9:2   284       BEGIN
1    9:3   284         EXIT(DELFUN)
1    9:2   288         END;
1    9:1   288     J:=0;
1    9:1   292     REPEAT
1    9:2   292       TEMP:=NPAC;
1    9:2   309       J:=J+1;
1    9:2   317       IF (CORE[J] DIV 100)=(TEMP*10000+NOBJECTIVE*100+I) THEN
1    9:3   392         REMOVE;
1    9:1   394       UNTIL (J=NCORELAST)
1    9:0   401   END;
1    9:0   420
```

DELFUN asks analyst which functional purpose he/she wishes to remove and removes the functional purpose and its component characteristics.

```
1   39:D     1 (*$P*)PROCEDURE FPUR4;
1   39:0     0   BEGIN
1   39:1     0      GOTOXY(0,16);
1   39:1     5      WRITELN(CHR(11));
1   39:1    23      REPEAT
1   39:2    23        WRITELN('You have chosen to reword a functional purpose');
1   39:2    89        WRITELN(' ');
1   39:2   107        WRITELN('Which one do you want to reword (Type 0 to reconsider)? ');
1   39:2   183        HELP:=15;
1   39:2   187        KEYN;
1   39:2   189        IF I=999 THEN
1   39:3   198          FPUR;
1   39:2   200        IF (I<0) OR (I>20) THEN
1   39:3   213          WRITELN('PLEASE TYPE AN INTEGER BETWEEN 0 AND 20');
1   39:1   272        UNTIL (I>=0) AND (I<21);
1   39:1   285      IF I=0 THEN
1   39:2   292        EXIT(FPUR4);
1   39:1   296      IF SCRATCH[I]='' THEN
1   39:2   316        BEGIN
1   39:3   316          WRITELN(NSCRATCH[I],' DOES NOT EXIST');
1   39:3   374          EXIT(FPUR4);
1   39:2   378          END;
1   39:1   378      GOTOXY(0,16);
1   39:1   383      WRITELN(CHR(11));
1   39:1   401      WRITELN('Please reword (80 characters available) the functional purpose');
1   39:1   483      WRITE(CHR(13));
1   39:1   493      PREFIXF;
1   39:1   495      INLINE;
1   39:1   497      IF SCRATCH[I]='' THEN
1   39:2   517        EXIT(FPUR4);
1   39:1   521      NSCRATCH[I]:=I;
1   39:1   538      DATANODE^.NTAXA[1]:=NFAC;
1   39:1   553      DATANODE^.NTAXA[2]:=NOBJECTIVE;
1   39:1   568      DATANODE^.NTAXA[3]:=NSCRATCH[I];
1   39:1   594      DATANODE^.NTAXA[4]:=0;
1   39:1   607      DATANODE^.TAXA:=SCRATCH[I];
1   39:1   627      TEMP:=NFAC;
1   39:1   644      TSCR:=NOBJECTIVE;
1   39:1   661      TEMP:=TEMP*1000000+TSCR*10000+NSCRATCH[I]*100+0;
1   39:1   760      J:=0;
1   39:1   764      REPEAT
1   39:2   764        J:=J+1;
1   39:1   772        UNTIL TEMP=CORE[J];
1   39:1   803      WRITELN('OK');
1   39:1   825      SEEK(DATANODE,J);
1   39:1   836      PUT(DATANODE);
1   39:0   844     END;
1   39:0   860
```

FPUR4 asks the analyst which functional purpose to reword and it asks them to reword the functional purpose.

```
852   1   40:D      1 {$$P$}PROCEDURE FPUR5;
853   1   40:0      0    BEGIN
854   1   40:1      0      WRITELN('Please be certain that the printer is ON and ONLINE!!!!!');
855   1   40:1     76      NPRINT:=2;
856   1   40:1     80      PRINTSCRN;
857   1   40:0     82      END;
858   1   40:0     94
```

FPUR5 calls PRINTSCRN to print the entire contents of the functional purpose display.

```
1   41:D     1 (*$P*)PROCEDURE FPUR7;
1   41:0     0    BEGIN
1   41:1     0      ASPECTS;
1   41:1     2      OBJECTIVES;
1   41:0     4      END;
1   41:0    16
```

FPUR7 calls ASPECTS and OBJECTIVES so the analyst can specify a new aspect.

```
1 42:D     1 (*$P*)PROCEDURE SELECTFPS;
1 42:0     0   BEGIN
1 42:1     0     GOTOXY(0,16);
1 42:1     5     WRITELN(CHR(11));
1 42:1    23     WRITE('You may perform any of the following procedures:',chr(13),
1 42:1    93 '  1. Analyze chararacteristics          2. Specify new functional purposes',chr(13),
1 42:1   187 '  3. Remove a functional purpose        4. Reword a functional purpose',chr(13),
1 42:1   277 '  5. Print these functional purposes   6. Analyze a different objective',chr(13),
1 42:1   369 '  7. Analyze a different aspect         8. Select a different analytic proc.',chr(13),
1 42:1   465 'Please select one: ');
1 42:1   496     REPEAT
1 42:2   496       HELP:=15;
1 42:2   500       KEYN;
1 42:2   502       IF I=999 THEN
1 42:3   511         FPUR;
1 42:2   513       IF (I<1) OR (I>8) THEN
1 42:3   526         WRITELN('Please type an integer between 1 and 8');
1 42:1   584       UNTIL (I>=1) AND (I<=8);
1 42:1   597     GOTOXY(0,16);
1 42:1   602     WRITE(CHR(11));
1 42:1   612     CASE I OF
1 42:1   617       1: FPUR1;
1 42:1   621       2: FUNCCREATE;
1 42:1   625       3: DELFUN;
1 42:1   629       4: FPUR4;
1 42:1   633       5: FPUR5;
1 42:1   637       6: OBJECTIVES;
1 42:1   641       7: FPUR7;
1 42:1   645       8: OBJ7; (*YES, IT IS OK*)
1 42:1   649       END;
1 42:0   672     END;
1 42:0   686
```

SELECTFPS displays analytic options available to the analyst at the bottom of the functional purposes page.

```
1   15:D     1 (*$P*)PROCEDURE FPUR;
1   15:0     0   BEGIN
1   15:1     0     NSCREEN:=2;
1   15:1     4     TOPSCREEN;
1   15:1     6     WRITE('Functional purposes--');
1   15:1    39     PREFIXF;
1   15:1    41     FOR J:=1 TO 20 DO
1   15:2    55       BEGIN
1   15:3    55         SCRATCH[J]:='';
1   15:3    73         NSCRATCH[J]:=J
1   15:2    86         END;
1   15:1   100     FOR I:=1 TO NCORELAST DO
1   15:2   116       BEGIN
1   15:3   116         IF CORE[I] DIV 10000=NPAC*100+NOBJECTIVE THEN
1   15:4   165           BEGIN
1   15:5   165             IF CORE[I]-CORE[I] DIV 10000 * 10000 <> 0 THEN
1   15:6   239               BEGIN
1   15:7   239                 IF CORE[I] - CORE[I] DIV 100 * 100 = 0 THEN
1   15:8   309                   BEGIN
1   15:9   309                     SEEK(DATANODE,I);
1   15:9   320                     GET (DATANODE);
1   15:9   328                     J:=DATANODE^.NTAXA[3];
1   15:9   343                     NSCRATCH[J]:=DATANODE^.NTAXA[3];
1   15:9   369                     SCRATCH[J]:=DATANODE^.TAXA;
1   15:8   389                     END;
1   15:6   389                 END;
1   15:4   389             END;
1   15:2   389         END;
1   15:1   399       DISPSCRATCH;
1   15:1   401       IF NDATA=0 THEN
1   15:2   408         WRITELN('I do not have any functional purposes for objective number',
1   15:2   478                NOBJECTIVE,' at the present time');
1   15:1   530       SELECTFPS;
1   15:1   532       FPUR;
1   15:0   534     END;
1   15:0   556
```

FPUR governs the primary display of functional purposes.

```
1    5:D      1 (*$P*)PROCEDURE PCHARCREATE;
1    5:0      0   BEGIN
1    5:1      0     FOR J:=1 TO 20 DO
1    5:2     14       BEGIN
1    5:3     14         SCRATCH[J]:='';
1    5:3     32         NSCRATCH[J]:=J
1    5:2     45         END;
1    5:0     59     END;
1    5:0     74
```

PCHARCREATE clears the STRATCH array which is used in producing the body of the display for objectives, functional purposes and characteristics.

```
1    8:D     1 (*$P*)PROCEDURE INDEX;
1    8:0     0   BEGIN
1    8:1     0     I:=0;
1    8:1     4     FOR J:=20 DOWNTO 1 DO
1    8:2    18       BEGIN
1    8:3    18         IF(SCRATCH[J]='') THEN
1    8:4    38           I:=J;
1    8:2    44         END;
1    8:1    54       IF I=0 THEN
1    8:2    61         BEGIN
1    8:3    61           WRITELN('All 20 indexes are currently in use!!');
1    8:3   118           ANYKEY;
1    8:2   120           END;
1    8:0   120       END;
1    8:0   134
1    8:0   134  (*$I #5:PERFITEM3.TEXT*)
```

INDEX determines whether there are 20 objectives for a given aspect, 20 functional purposes for a given objective or 20 characteristics for a given functional purpose before allowing new performance items to be added. If the maximum are in use, additional items cannot be added.

```
 1   43:D     1 (*$P*)PROCEDURE OBJCNODE;
 1   43:0     0   BEGIN
 1   43:1     0     DATANODE^.NTAXA[1]:=NPAC;
 1   43:1    15     DATANODE^.NTAXA[2]:=NSCRATCH[I];
 1   43:1    41     DATANODE^.NTAXA[3]:=0;
 1   43:1    54     DATANODE^.NTAXA[4]:=0;
 1   43:1    67     DATANODE^.TAXA:=SCRATCH[I];
 1   43:1    87     REPEAT
 1   43:2    87       BEGIN
 1   43:3    87         J:=TRUNC(CORELAST);
 1   43:3   100         IF J>=300 THEN
 1   43:4   109           BEGIN
 1   43:5   109             WRITELN('*** ERROR -- YOUR DATA SET ALREADY CONTAINS 300 PERFORMANCE
                            ITEMS! ***');
 1   43:5   199             WRITELN('***          THUS, THIS ITEM WAS NOT ADDED TO DATA SET ***');
 1   43:5   277             ANYKEY;
 1   43:5   279             EXIT(OBJCREATE);
 1   43:4   283             END;
 1   43:3   283         CORELAST:=CORELAST+1;
 1   43:3   308         J:=J+1;
 1   43:3   316         EII:=CORE[J] DIV 1000000;
 1   43:3   373         NCORELAST:=J;
 1   43:2   379         END;
 1   43:1   379       UNTIL EII =0;
 1   43:1   397     TEMP:=NPAC;
 1   43:1   414     TSCR:=NSCRATCH[I];
 1   43:1   442     CORE[J]:=TEMP*1000000+TSCR*10000+0*100+0;
 1   43:1   540     SEEK(DATANODE,J);
 1   43:1   551     PUT(DATANODE);
 1   43:0   559     END;
 1   43:0   576
```

OBJCNODE adds a new objective to the performance item list.

```
 1    44:D     1 (*$P*)PROCEDURE OBJ1;
 1    44:0     0   BEGIN
 1    44:1     0      WRITE('Which objective would you like to analyze (type 0 to reconsider)?');
 1    44:1    77        REPEAT
 1    44:2    77          HELP:=23;
 1    44:2    81          KEYN;
 1    44:2    83          IF I=999 THEN
 1    44:3    92            OBJECTIVES;
 1    44:2    94          IF(I<0)OR(I>20)THEN
 1    44:3   107            WRITELN('PLEASE INPUT AN INTEGER BETWEEN 0 AND 20');
 1    44:2   167          IF I>0 THEN
 1    44:3   174            IF SCRATCH[I]='' THEN
 1    44:4   194              BEGIN
 1    44:5   194                WRITELN(I,' DOES NOT EXIST AT PRESENT');
 1    44:5   252                WRITELN('PLEASE TRY ANOTHER OBJECTIVE');
 1    44:5   300                I:=25;
 1    44:4   304                END;
 1    44:1   304          UNTIL (I>=0) AND (I<20);
 1    44:1   317      IF (I>0) AND (I<21) THEN
 1    44:2   330        BEGIN
 1    44:3   330          NOBJECTIVE:=NSCRATCH[I];
 1    44:3   347          XOBJECTIVE:=SCRATCH[I];
 1    44:3   365          FPUR;
 1    44:2   367          END;
 1    44:0   367    END;
 1    44:0   384
```

OBJ1 asks analyst which objective he wishes to analyze when he requests to analyze functional purposes.

```
1013 1 12:D    (*$P*)PROCEDURE OBJCREATE;
1014 1 12:0    0  BEGIN
1015 1 12:1    0    REPEAT
1016 1 12:2    0      NSCREEN:=1;
1017 1 12:2    4      TOPSCREEN;
1018 1 12:2    6      WRITELN('You have chosen to create a new objective.');
1019 1 12:2   68      WRITELN(' ');
1020 1 12:2   86      INDEX;
1021 1 12:2   88      IF I=0 THEN
1022 1 12:3   95        EXIT(OBJCREATE);
1023 1 12:2   99      WRITELN('Please specify (80 additional characters available) the new objective',
1024 1 12:2  180              CHR(13),' within the ',PAC,' aspect of the ',CURSUB,' subsystem');
1025 1 12:2  294      WRITE(CHR(13));
1026 1 12:2  304      PREFIXO;
1027 1 12:2  306      INLINE;
1028 1 12:2  308      IF SCRATCH[I]='' THEN
1029 1 12:3  328        EXIT(OBJCREATE);
1030 1 12:2  332      NSCRATCH[I]:=I;
1031 1 12:2  349      OBJCNODE;
1032 1 12:2  351      WRITELN('OK');
1033 1 12:2  373      WRITELN('It will, at some time, be necessary to add functional purposes and',chr(13),
1034 1 12:2  461              '  characteristics to this objective');
1035 1 12:2  516      PREPKEY(17,'Would you like to specify functional purposes at this time?');
1036 1 12:2  581      IF ANS='Y' THEN
1037 1 12:3  588        BEGIN
1038 1 12:4  588          XOBJECTIVE:=SCRATCH[I];
1039 1 12:4  606          NOBJECTIVE:=NSCRATCH[I];
1040 1 12:4  623          PCHARCREATE; (*YES, ITS OK*)
1041 1 12:4  625          FUNCCREATE;
1042 1 12:4  627          EXIT(OBJCREATE);
1043 1 12:3  631          END;
1044 1 12:2  631      PREPKEY(23,'Would you like to specify more objectives?');
1045 1 12:2  679      IF ANS='Y' THEN
1046 1 12:3  686        OK:=FALSE
1047 1 12:2  686        ELSE
1048 1 12:3  692          EXIT(OBJCREATE);
1049 1 12:1  696      UNTIL OK;
1050 1 12:0  701    END;
1051 1 12:0  716
```

OBJCREATE accepts a new performance item and calls OBJCNODE to add it to the list of performance items.

```
 1   19:D     1 (*$P*)PROCEDURE REMOVE;
 1   19:0     0   BEGIN
 1   19:1     0     DATANODE^.NTAXA[1]:=0;
 1   19:1    13     DATANODE^.NTAXA[2]:=0;
 1   19:1    26     DATANODE^.NTAXA[3]:=0;
 1   19:1    39     DATANODE^.NTAXA[4]:=0;
 1   19:1    52     DATANODE^.TAXA:='';
 1   19:1    62     SEEK(DATANODE,J);
 1   19:1    73     PUT(DATANODE);
 1   19:1    81     CORE[J]:=0;
 1   19:0   108     END;
 1   19:0   120
```

REMOVE removes unwanted performance items from the performance item file.

```
 1   45:D      1 {$$P$}PROCEDURE DELOBJ;
 1   45:0      0   BEGIN
 1   45:1      0     REPEAT
 1   45:2      0       GOTOXY(0,16);
 1   45:2      5       WRITE(chr(11),'Which one do you wish to remove (Type 0 to reconsider):');
 1   45:2     82       KEYN;
 1   45:2     84       IF (I<0) OR (I>20) THEN
 1   45:3     97         WRITELN('PLEASE INPUT AN INTEGER BETWEEN 0 AND 20');
 1   45:1    157       UNTIL (I>=0)AND(I<21);
 1   45:1    170     IF I=0 THEN
 1   45:2    177       BEGIN
 1   45:3    177         EXIT(DELOBJ);
 1   45:2    181         END;
 1   45:1    181     PREPKEY(23,'Do you really want to remove this objective & component functional
                     purposes?');
 1   45:1    263     IF ANS='N' THEN
 1   45:2    270       BEGIN
 1   45:3    270         EXIT(DELOBJ)
 1   45:2    274         END;
 1   45:1    274     J:=0;
 1   45:1    278     REPEAT
 1   45:2    278       J:=J+1;
 1   45:2    286       IF (CORE[J] DIV 10000)=(NPAC*100+I) THEN
 1   45:3    335         REMOVE;
 1   45:1    337       UNTIL (J=NCORELAST)
 1   45:0    344   END;
 1   45:0    362
```

DELOBJ asks analyst which objective he/she wishes to remove and calls REMOVE to remove the objective and component functional purposes and characteristics.

```
1    46:D      1 (*$P*)PROCEDURE OBJ4;
1    46:0      0   BEGIN
1    46:1      0     GOTOXY(0,16);
1    46:1      5     WRITE(CHR(11));
1    46:1     15     REPEAT;
1    46:2     15       WRITELN('You have chosen to reword an objective.');
1    46:2     74       WRITELN(' ');
1    46:2     92       WRITE('Which one do you want to reword (Type 0 to reconsider)? ');
1    46:2    160       KEYN;
1    46:2    162       IF (I<0) OR (I>20) THEN
1    46:3    175         WRITELN('PLEASE TYPE AN INTEGER BETWEEN 0 AND 20');
1    46:1    234       UNTIL (I>=0) AND (I<21);
1    46:1    247     GOTOXY(0,16);
1    46:1    252     WRITE(CHR(11));
1    46:1    262     IF I=0 THEN
1    46:2    269       EXIT(OBJ4);
1    46:1    273     IF (SCRATCH[I]='')THEN
1    46:2    293       BEGIN
1    46:3    293         WRITELN(NSCRATCH[I],' DOES NOT EXIST');
1    46:3    351         EXIT(OBJ4);
1    46:2    355         END;
1    46:1    355     WRITELN('Please reword (80 characters available) this objective.');
1    46:1    430     WRITE(CHR(13));
1    46:1    440     PREFIXO;
1    46:1    442     INLINE;
1    46:1    444     IF SCRATCH[I]='' THEN
1    46:2    464       EXIT(OBJCREATE);
1    46:1    468     NSCRATCH[I]:=I;
1    46:1    485     DATANODE^.NTAXA[1]:=NPAC;
1    46:1    500     DATANODE^.NTAXA[2]:=NSCRATCH[I];
1    46:1    526     DATANODE^.NTAXA[3]:=0;
1    46:1    539     DATANODE^.NTAXA[4]:=0;
1    46:1    552     DATANODE^.TAXA:=SCRATCH[I];
1    46:1    572     TEMP:=NPAC;
1    46:1    589     TSCR:=NSCRATCH[I];
1    46:1    617     TEMP:=TEMP*1000000+TSCR*10000+0*100+0;
1    46:1    703     J:=0;
1    46:1    707     REPEAT
1    46:2    707       J:=J+1;
1    46:1    715       UNTIL TEMP=CORE[J];
1    46:1    746     WRITELN('OK');
1    46:1    768     SEEK(DATANODE,J);
1    46:1    779     PUT(DATANODE);
1    46:0    787     END;
1    46:0    804
```

OBJ4 asks analyst which objective he/she wishes to reword and asks him/her to reword it.

```
1    47:D     1 {$$P$}PROCEDURE OBJ5;
1    47:0     0   BEGIN
1    47:1     0     WRITELN('Please be certain that the printer is ON and ONLINE!!!!!');
1    47:1    76     NPRINT:=1;
1    47:1    80     PRINTSCRN;
1    47:0    82     END;
1    47:0    94
```

OBJ5 calls PRINTSCREEN to print the contents of the objectives display screen.

```
1142   1    14:D     1 (*$P*)PROCEDURE OBJ7;
1143   1    14:0     0   BEGIN
1144   1    14:1     0     CORECLOSE;
1145   1    14:1     2     BRANCHOUT;
1146   1    14:1     4     SETCHAIN('GREETING');
1147   1    14:1    18     EXIT(PROGRAM);
1148   1    14:0    22     END;
1149   1    14:0    34
```

OBJ7 transfers control to the GREETING program.

```
  1   11:D     1 (*$P*)PROCEDURE DISPSCRATCH;
  1   11:0     0   BEGIN
  1   11:1     0     NDATA:=0;
  1   11:1     4     PC:=1;
  1   11:1     8     FOR K:=1 TO 20 DO
  1   11:2    22       BEGIN
  1   11:3    22         IF SCRATCH[K]<>'' THEN
  1   11:4    42           BEGIN
  1   11:5    42             IF PC>=10 THEN
  1   11:6    49               BEGIN
  1   11:7    49                 PC:=1;
  1   11:7    53                 ANYKEY;
  1   11:7    55                 GOTOXY(0,5);
  1   11:7    60                 WRITE(CHR(11));
  1   11:6    70                 END;
  1   11:5    70             NDATA:=1;
  1   11:5    74             LLENGTH:=72;
  1   11:5    78             LINE:=SCRATCH[K];
  1   11:5    96             WRITE('  ',NSCRATCH[K],'.  ');
  1   11:5   148             SHOWALINE;
  1   11:5   150             WRITELN(' ');
  1   11:5   168             PC:=PC+1;
  1   11:4   176             END;
  1   11:2   176         END;
  1   11:0   186     END;
  1   11:0   204
```

DISPSCRATCH displays the performance items in the body of the main displays.

```
1   48:D     1 (*$P*)PROCEDURE SELECTOBJECTIVES;
1   48:0     0   BEGIN
1   48:1     0     GOTOXY(0,16);
1   48:1     5     WRITE(CHR(11));
1   48:1    15     WRITE('You may perform any of the following procedures:',chr(13),
1   48:1    85 '  1. Analyze functional purposes      2. Specify a new objective',chr(13),
1   48:1   171 '  3. Remove an objective              4. Reword an objective',chr(13),
1   48:1   253 '  5. Print these objectives           6. Analyze a different aspect',chr(13),
1   48:1   342 '  7. Select a different analytic proc.  ',chr(13),
1   48:1   404 'Please select one: ');
1   48:1   435     REPEAT
1   48:2   435       HELP:=23;
1   48:2   439       KEYN;
1   48:2   441       IF I=999 THEN
1   48:3   450         OBJECTIVES;
1   48:2   452       IF (I<1) OR (I>7) THEN
1   48:3   465         WRITELN('Please type an integer between 1 and 7');
1   48:1   523       UNTIL (I>=1) AND (I<=7);
1   48:1   536     GOTOXY(0,16);
1   48:1   541     WRITE(CHR(11));
1   48:1   551     CASE I OF
1   48:1   556       1: OBJ1;
1   48:1   560       2: OBJCREATE;
1   48:1   564       3: DELOBJ;
1   48:1   568       4: OBJ4;
1   48:1   572       5: OBJ5;
1   48:1   576       6: ASPECTS;
1   48:1   580       7: OBJ7;
1   48:1   584       END;
1   48:0   606     END;
1   48:0   620
```

SELECTOBJECTIVES prints the menu of the analytic processes available at the objectives level on the bottom of the display screen.

```
1     6:D      1 (*$P*)PROCEDURE OBJECTIVES;
1     6:0      0   BEGIN
1     6:1      0     NSCREEN:=1;
1     6:1      4     TOPSCREEN;
1     6:1      6     WRITE('Objectives--');
1     6:1     30     PREFIXO;
1     6:1     32     FOR J:=1 TO 20 DO
1     6:2     46       BEGIN
1     6:3     46         SCRATCH[J]:='';
1     6:3     64         NSCRATCH[J]:=J
1     6:2     77         END;
1     6:1     91     FOR I:=1 TO NCORELAST DO
1     6:2    107       BEGIN
1     6:3    107         IF CORE[I] DIV 1000000=NPAC THEN
1     6:4    168           BEGIN
1     6:5    168             IF CORE[I] - CORE[I] DIV 10000 * 10000 = 0 THEN
1     6:6    242               BEGIN
1     6:7    242                 SEEK(DATANODE,I);
1     6:7    253                 GET (DATANODE);
1     6:7    261                 J:=DATANODE^.NTAXA[2];
1     6:7    276                 NSCRATCH[J]:=DATANODE^.NTAXA[2];
1     6:7    302                 SCRATCH[J]:=DATANODE^.TAXA;
1     6:6    322                 END;
1     6:4    322             END;
1     6:2    322          END;
1     6:1    332       DISPSCRATCH;
1     6:1    334       IF NDATA=0 THEN
1     6:2    341         WRITELN('I have no objectives for aspect ',PAC,' at this time');
1     6:1    430       SELECTOBJECTIVES;
1     6:1    432       OBJECTIVES;
1     6:0    434       END;
1     6:0    454
1     6:0    454 (*$I #5:PERFITEM3.TEXT*)
1     6:0    454
```

OBJECTIVES governs the overall main display of the objectives.

```
1   49:D     1 (*$P*)PROCEDURE CH1;
1   49:0     0   BEGIN
1   49:1     0     NSCREEN:=3;
1   49:1     4     TOPSCREEN;
1   49:1     6     WRITELN('You have chosen to create a new characteristic.');
1   49:1    73     WRITELN(' ');
1   49:1    91     INDEX;
1   49:0    93     END;
1   49:0   106
```

CH1 calls index to be certain that there is room to add the desired characteristics to the list of performance items.

```
 1 3:D     1 (**P*)PROCEDURE CHARCREATE;
 1 3:0     0   BEGIN
 1 3:1     0     REPEAT
 1 3:2     0       CH1;
 1 3:2     2       IF I=0 THEN
 1 3:3     9         EXIT (CHARCREATE);
 1 3:2    13       WRITELN('Please specify (80 additional characters available) the new
                   characteristic?');
 1 3:2   108       WRITE(CHR(13));
 1 3:2   118       PREFIXC;
 1 3:2   120       INLINE;
 1 3:2   122       IF SCRATCH[I]='' THEN
 1 3:3   142         EXIT(CHARCREATE);
 1 3:2   146       NSCRATCH[I]:=I;
 1 3:2   163       WRITELN('OK');
 1 3:2   185       DATANODE^.NTAXA[1]:=NPAC;
 1 3:2   200       DATANODE^.NTAXA[2]:=NOBJECTIVE;
 1 3:2   215       DATANODE^.NTAXA[3]:=NFUNPUR;
 1 3:2   230       DATANODE^.NTAXA[4]:=NSCRATCH[I];
 1 3:2   256       DATANODE^.TAXA:=SCRATCH[I];
 1 3:2   276       REPEAT
 1 3:3   276         BEGIN
 1 3:4   276           J:=TRUNC(CORELAST);
 1 3:4   289           IF J>=300 THEN
 1 3:5   298             BEGIN
 1 3:6   298               WRITELN('*** ERROR -- YOUR DATA SET ALREADY CONTAINS 300 PERFORMANCE
                           ITEMS! ***');
 1 3:6   388               WRITELN('***          THUS, THIS ITEM WAS NOT ADDED TO DATA SET ***');
 1 3:6   466               ANYKEY;
 1 3:6   468               EXIT(FUNCCREATE);
 1 3:5   472               END;
 1 3:4   472           CORELAST:=CORELAST+1;
 1 3:4   497           J:=J+1;
 1 3:4   505           EII:=CORE[J] DIV 1000000;
 1 3:4   562           NCORELAST:=J;
 1 3:3   568           END;
 1 3:2   568         UNTIL EII =0;
 1 3:2   586       TEMP:=NPAC;
 1 3:2   603       TSCR:=NOBJECTIVE;
 1 3:2   620       CORE[J]:=TEMP*1000000+TSCR*10000+NFUNPUR*100+NSCRATCH[I];
```

CHARCREATE is the main routine to handle creating new characteristics.

```
1    3:2   733       SEEK(DATANODE,J);
1    3:2   744       PUT(DATANODE);
1    3:2   752       PREPKEY(12,'Would you like to specify more characteristics?');
1    3:2   805       IF ANS='Y' THEN
1    3:3   812         OK:=FALSE
1    3:2   812         ELSE
1    3:3   818           EXIT(CHARCREATE);
1    3:1   822       UNTIL OK;
1    3:0   827     END;
1    3:0   846
```

See previous page for program description.

```
1   10:D     1 (*$P*)PROCEDURE DELCAR;
1   10:0     0   BEGIN
1   10:1     0     OVER:=FALSE;
1   10:1     4     REPEAT
1   10:2     4       WRITE('Which one (Type 0 to reconsider):');
1   10:2    49       HELP:=12;
1   10:2    53       KEYN;
1   10:2    55       IF I=999 THEN
1   10:3    64         CHARACTERISTICS;
1   10:2    66       IF (I<0) OR (I>20) THEN
1   10:3    79         WRITELN('PLEASE INPUT AN INTEGER BETWEEN 0 AND 20');
1   10:1   139       UNTIL (I>=0)AND(I<21);
1   10:1   152     IF I=0 THEN
1   10:2   159       BEGIN
1   10:3   159         EXIT(DELCAR);
1   10:2   163         END;
1   10:1   163     PREPKEY(12,'Do you really want to remove this characteristic?');
1   10:1   218     J:=0;
1   10:1   222     IF ANS='N' THEN
1   10:2   229       BEGIN
1   10:3   229         EXIT(DELFUN)
1   10:2   233         END
1   10:1   233       ELSE
1   10:2   235         REPEAT
1   10:3   235           TEMP:=NPAC;
1   10:3   252           TSCR:=NOBJECTIVE;
1   10:3   269           J:=J+1;
1   10:3   277           IF CORE[J]=TEMP*1000000+TSCR*10000+NFUNPUR*100+I THEN
1   10:4   382             REMOVE;
1   10:2   384           UNTIL (J=NCORELAST)
1   10:0   391   END;
1   10:0   412
```

DELCAR asks analyst which characteristic he/she wishes to delete and calls REMOVE to actually remove the characteristic.

```
 1   50:D     1 (*$P*)PROCEDURE CHAR3;
 1   50:0     0   BEGIN
 1   50:1     0     GOTOXY(0,16);
 1   50:1     5     WRITE(CHR(11));
 1   50:1    15      REPEAT
 1   50:2    15       WRITELN('You have chosen to reword a characteristic.');
 1   50:2    78       WRITELN(' ');
 1   50:2    96       HELP:=12;
 1   50:2   100       WRITE('Which one do you want to reword (Type 0 to reconsider)? ');
 1   50:2   168       KEYN;
 1   50:2   170       IF I=999 THEN
 1   50:3   179         CHARACTERISTICS;
 1   50:2   181       IF (I<0) OR (I>20) THEN
 1   50:3   194         WRITELN('PLEASE TYPE AN INTEGER BETWEEN 0 AND 20');
 1   50:1   253       UNTIL (I>=0) AND (I<21);
 1   50:1   266     IF I=0 THEN
 1   50:2   273       EXIT(CHAR3);
 1   50:1   277     IF SCRATCH[I]='' THEN
 1   50:2   297       BEGIN
 1   50:3   297         WRITELN(NSCRATCH[I],' DOES NOT EXIST');
 1   50:3   355         EXIT(CHAR3);
 1   50:2   359         END;
 1   50:1   359     GOTOXY(0,16);
 1   50:1   364     WRITE(CHR(11));
 1   50:1   374     WRITELN('Please reword (80 characters available) this characteristic?');
 1   50:1   454     WRITE(CHR(13));
 1   50:1   464     PREFIXC;
 1   50:1   466     INLINE;
 1   50:1   468     IF SCRATCH[I]='' THEN
 1   50:2   488       EXIT(CHAR3);
 1   50:1   492     NSCRATCH[I]:=I;
 1   50:1   509     DATANODE^.NTAXA[1]:=NPAC;
 1   50:1   524     DATANODE^.NTAXA[2]:=NOBJECTIVE;
 1   50:1   539     DATANODE^.NTAXA[3]:=NFUNFUR;
 1   50:1   554     DATANODE^.NTAXA[4]:=NSCRATCH[I];
 1   50:1   580     DATANODE^.TAXA:=SCRATCH[I];
 1   50:1   600     TEMP:=NPAC;
 1   50:1   617     TSCR:=NOBJECTIVE;
 1   50:1   634     TEMP:=TEMP*1000000+TSCR*10000+NFUNFUR*100+NSCRATCH[I];
 1   50:1   735     J:=0;
 1   50:1   739     REPEAT
 1   50:2   739       J:=J+1;;
 1   50:1   747       UNTIL TEMP=CORE[J];
 1   50:1   778     WRITELN('OK');
 1   50:1   800     SEEK(DATANODE,J);
 1   50:1   811     PUT(DATANODE);
 1   50:0   819     END;
 1   50:0   836
```

CHAR3 asks analyst which characteristic he/she wishes to reword. It then asks him/her for the new wording of the characteristic.

```
1   51:D     1 (*$P*)PROCEDURE CHAR4;
1   51:0     0   BEGIN
1   51:1     0     WRITELN('Please be certain that the printer is ON and ONLINE!!!!!');
1   51:1    76     NPRINT:=3;
1   51:1    80     PRINTSCRN;
1   51:0    82     END;
1   51:0    94
```

CHAR4 calls PRINTSCRN to print the main screen for the characteristics level performance items.

```
1   52:D     1 (*$P*)PROCEDURE CHAR6;
1   52:0     0   BEGIN
1   52:1     0     OBJECTIVES;
1   52:1     2     FPUR;
1   52:0     4     END;
1   52:0    16
```

CHAR6 enables analyst to specify a different objective (and consequently, a different functional purpose).

```
1   53:D     1 (*$P*)PROCEDURE CHAR7;
1   53:0     0   BEGIN
1   53:1     0     ASPECTS;
1   53:1     2     OBJECTIVES;
1   53:1     4     FPUR;
1   53:0     6     END;
1   53:0    18
```

CHAR7 allows analyst to specify a different aspect (and consequently, a different objective and functional purpose).

```
1 54:D     1 (*$P*)PROCEDURE SELECTCHARACTERISTICS;
1 54:0     0   BEGIN
1 54:1     0     GOTOXY(0,16);
1 54:1     5     WRITE(CHR(11));
1 54:1    15     WRITE('You may perform any of the following procedures:',chr(13),
1 54:1    85 '  1. Specify new characteristics      2. Remove a characteristic',chr(13),
1 54:1   171 '  3. Reword a characteristic          4. Print these characteristics',chr(13),
1 54:1   261 '  5. Analyze a different func. purp.  6. Analyze a different objective',chr(13),
1 54:1   353 '  7. Analyze a different aspect       8. Select a different analytic proc.',chr(13),
1 54:1   449 'Please select one: ');
1 54:1   480     REPEAT
1 54:2   480       HELP:=12;
1 54:2   484       KEYN;
1 54:2   486       IF I=999 THEN
1 54:3   495         CHARACTERISTICS;
1 54:2   497       IF (I<1) OR (I>8) THEN
1 54:3   510         WRITELN('Please type an integer between 1 and 8');
1 54:1   568       UNTIL (I>=1) AND (I<=8);
1 54:1   581     GOTOXY(0,16);
1 54:1   586     WRITE(CHR(11));
1 54:1   596     CASE I OF
1 54:1   601       1: CHARCREATE;
1 54:1   605       2: DELCAR;
1 54:1   609       3: CHAR3;
1 54:1   613       4: CHAR4;
1 54:1   617       5: FPUR;
1 54:1   621       6: CHAR6;
1 54:1   625       7: CHAR7;
1 54:1   629       8: OBJ7; (*YES, IT IS OK*)
1 54:1   633       END;
1 54:0   656     END;
1 54:0   670
```

SELECTCHARACTERISTICS displays the menu of procedures to be performed with characteristics level taxa.

```
1    4:D     1 (*$P*)PROCEDURE CHARACTERISTICS;
1    4:0     0   BEGIN
1    4:1     0     NSCREEN:=3;
1    4:1     4     TOPSCREEN;
1    4:1     6     WRITE('Characteristics--');
1    4:1    35     PREFIXC;
1    4:1    37     FOR J:=1 TO 20 DO
1    4:2    51       BEGIN
1    4:3    51         SCRATCH[J]:='';
1    4:3    69         NSCRATCH[J]:=J
1    4:2    82         END;
1    4:1    96     FOR I:=1 TO NCORELAST DO
1    4:2   112       BEGIN
1    4:3   112         TEMP:=NPAC;
1    4:3   129         IF CORE[I] DIV 100=TEMP*10000+NOBJECTIVE*100+NFUNPUR THEN
1    4:4   204           BEGIN
1    4:5   204             IF CORE[I] - CORE[I] DIV 100 * 100 <> 0 THEN
1    4:6   274               BEGIN
1    4:7   274                 SEEK(DATANODE,I);
1    4:7   285                 GET (DATANODE);
1    4:7   293                 J:=DATANODE^.NTAXA[4];
1    4:7   308                 NSCRATCH[J]:=DATANODE^.NTAXA[4];
1    4:7   334                 SCRATCH[J]:=DATANODE^.TAXA;
1    4:6   354                 END;
1    4:4   354             END;
1    4:2   354         END;
1    4:1   364     DISPSCRATCH;
1    4:1   366     IF NDATA=0 THEN
1    4:2   373       WRITELN('No characteristics are available at the present time');
1    4:1   445     SELECTCHARACTERISTICS;
1    4:1   447     CHARACTERISTICS;
1    4:0   449     END;
1    4:0   470
```

CHARACTERISTICS governs the characteristics display functions.

```
 1    2:D     1 (*$P*)PROCEDURE CORECLOSE;
 1    2:0     0   BEGIN
 1    2:1     0     RESET(COREFILE,NAMECOREFILE);
 1    2:1    13     FOR I:=1 TO 300 DO
 1    2:2    29       BEGIN
 1    2:3    29         COREFILE^:=CORE[I];
 1    2:3    57         PUT (COREFILE)
 1    2:2    65         END;
 1    2:1    75     COREFILE^:=CORELAST;
 1    2:1    91     PUT(COREFILE);
 1    2:1    99     CLOSE(COREFILE);
 1    2:0   108     END;
 1    2:0   122
```

CORECLOSE saves the index to the performance items on disk.

```
1 1:0    0   (*$P*)BEGIN
1 1:0    0     (*$N-*)
1 1:1    0     INLINECALL:=0;
1 1:1   66     BRANCHIN;
1 1:1   68     APMDSK:=CONCAT(COPY(CURSYS,1,2),COPY(CURSP,1,2),COPY(CURSUB,1,2),':');
1 1:1  157     NAMEHELPFILE:=CONCAT(APMDSK,'HELP');
1 1:1  193     NAMECOREFILE:=CONCAT(APMDSK,COPY(CURSYS,1,4),COPY(CURSP,1,4),COPY(CURSUB,1,4),'CO');
1 1:1  293     NAMEDATAFILE:=CONCAT(APMDSK,COPY(CURSYS,1,4),COPY(CURSP,1,4),COPY(CURSUB,1,4),'FI');
1 1:1  393     OPENCOREFILE;
1 1:1  395     ASPECTS;
1 1:1  397     OPENOBJFILE;
1 1:1  399     OBJECTIVES;
1 1:0  401     END.
```

MAIN PROGRAM for analyzing performance items.

MEASURES AND ATTRIBUTES PROGRAM (MEASATTR)

The measurement and attributes program allows the analyst to edit attributes and measures for each performance item, adding, rewording and deleting as appropriate.

```
 1    1:D     1 (*$L PRINTER: *)
 1    1:D     1 (*$S+*)
 1    1:D     1 (* Program to perform composition of attribute list*)
 1    1:D     1 (* Ronald G. Shapiro   Version 2.0         10/25/82*)
 1    1:D     1 Program Formattribute;
 1    1:D     3
28    1:D     3
28    2:D     1   PROCEDURE SETCHAIN(TYTLE:STRING);
28    3:D     1   PROCEDURE SETCVAL(VAL:STRING);
28    4:D     1   PROCEDURE GETCVAL(VAR VAL:STRING);
28    5:D     1   PROCEDURE SWAPON;
28    6:D     1   PROCEDURE SWAPOFF;
28    6:D     1
 1    1:D     1 Uses Chainstuff;
 1    1:D     3
```

These procedures are part of the Apple Computer's CHAINSTUFF library entry. The demonstration package uses only SETCHAIN which causes another program to be activated.

```
1    1:D     3 (*$P*)TYPE
1    1:D     3   PASSFILE =RECORD
1    1:D     3     CURSYS,CURSP,CURSUB,PAC:STRING[80];
1    1:D     3     NCURSYS,NCURSP,NCURSUB,NPAC,FLAG1,FLAG2,FLAG3:INTEGER;
1    1:D     3     END;
1    1:D     3
1    1:D     3   DATABASE =RECORD
1    1:D     3     NTAXA: ARRAY[1..4] OF INTEGER;
1    1:D     3     TAXA: STRING[80];
1    1:D     3     END;
1    1:D     3
1    1:D     3   FILEATTRIBUTES =RECORD
1    1:D     3     NDESCRIPTOR: ARRAY[1..6] OF INTEGER;
1    1:D     3     DESCRIPTOR: STRING[68];
1    1:D     3     END;
1    1:D     3
1    1:D     3   FILEMEASURES =RECORD
1    1:D     3     NDESCRIPTOR: ARRAY[1..6] OF INTEGER;
1    1:D     3     DESCRIPTOR: STRING[68];
1    1:D     3     END;
1    1:D     3
```

PASSFILE is used for interprogram communication (see GREETING listing).
FILEATTRIBUTES contains the attributes.
FILEMEASURES contains the measures.

```
 1    1:D     3 (*$P*)VAR
 1    1:D     3   PASSNODE:FILE OF PASSFILE;
 1    1:D   474   DATANODE:FILE OF DATABASE;
 1    1:D   819   COREFILE:FILE OF INTEGER[8];
 1    1:D  1122   ATTRIBUTES:FILE OF FILEATTRIBUTES;
 1    1:D  1463   ATTRFILE:FILE OF INTEGER[12];
 1    1:D  1767   MEASURES:FILE OF FILEMEASURES;
 1    1:D  2108   MEASFILE:FILE OF INTEGER[12];
 1    1:D  2412
 1    1:D  2412   CORE:ARRAY[1..300] OF INTEGER[8];
 1    1:D  3312   ATTRCORE:ARRAY[1..200] OF INTEGER[12];
 1    1:D  4112   MEASCORE:ARRAY[1..400] OF INTEGER[12];
 1    1:D  5712
 1    1:D  5712   SCRATCH:ARRAY [1..20] OF INTEGER;
 1    1:D  5732   ASPECT:ARRAY[1..5] OF STRING[14];
 1    1:D  5772   CORE2:ARRAY[1..300] OF INTEGER;
 1    1:D  6072   ATTRINDEX:ARRAY[1..20] OF INTEGER;
 1    1:D  6092   MEASINDEX:ARRAY[1..20] OF INTEGER;
 1    1:D  6112
 1    1:D  6112   XCHARAC,XFUNPUR,XOBJECTIVE,PAC,CURSYS,CURSP,CURSUB: STRING[80];
 1    1:D  6399   NCURMEASURES,NCURATTRIBUTE,NCURISSUE,NCHARAC,
 1    1:D  6399     NFUNPUR,NOBJECTIVE,NPAC,NCURSYS,NCURSP,NCURSUB: INTEGER;
 1    1:D  6409
 1    1:D  6409   NAMEATCORE,NAMEATTRIBUTES,NAMEMECORE,NAMEMEASURES: STRING[24];
 1    1:D  6461   CORENAME,DATANAME: STRING[24];
 1    1:D  6487   APMDSK:STRING[8];
 1    1:D  6492
 1    1:D  6492   TEMPL4,TEMP,TEMPL1,TEMPL2,TEMPL3,CORELAST: INTEGER[8];
 1    1:D  6510   TEMPX,ATTRLAST,MEASLAST:INTEGER[12];
 1    1:D  6522
 1    1:D  6522   NODE,INVERSE,HELP,NSCREEN,NPRINT:INTEGER;
 1    1:D  6527   NCORELAST,NATTRLAST,NMEASLAST:INTEGER;
 1    1:D  6530   NATTRIBUTES,NMEASURES,NUMEASURES:INTEGER;
 1    1:D  6533
 1    1:D  6533   INLINECALL,INDENT,LLENGTH,NLENGTH,PC,I,J,K,L,M,N,NATTR,NMEAS,DISPMCOUUNT,
                  DISPCOUNT,COUNT,TEMP2:IN
 1    1:D  6550
 1    1:D  6550   REFERENCED,LONGWAY,DONE,OVER,OK,SKIP,NONE:BOOLEAN;
 1    1:D  6557
 1    1:D  6557   ANSWER,LINE,REGLINE,LINER:STRING[80];
 1    1:D  6721
 1    1:D  6721   ANS,ANSHOLD: CHAR;
 1    1:D  6723
 1    1:D  6723   PRNT:TEXT;
 1    1:D  7024
 1    2:D     1 PROCEDURE BRANCHIN;FORWARD;
 1    3:D     1 PROCEDURE BRANCHOUT;FORWARD;
 1    4:D     1 PROCEDURE ANYKEY;FORWARD;
 1    4:D     1
```

These strings, arrays and variables are used by this program.

```
 7 1:D     1 (*$P*)SEGMENT PROCEDURE OPENATTRIBUTESFILE;
 7 1:0     0   BEGIN
 7 1:0     0     (*$I-*)
 7 1:1     0     RESET(ATTRIBUTES,NAMEATTRIBUTES);
 7 1:1    11     (*$I+*)
 7 1:1    11     I:=IORESULT;
 7 1:1    16     IF I<>0 THEN
 7 1:2    23       BEGIN
 7 1:3    23         WRITELN('Please bear with me while I create the attributes file on the disk');
 7 1:3   109         REWRITE(ATTRIBUTES,NAMEATTRIBUTES);
 7 1:3   122         FOR I:=1 TO NATTRIBUTES DO
 7 1:4   138           BEGIN
 7 1:5   138             SEEK(ATTRIBUTES,I);
 7 1:5   149             FOR J:=1 TO 6 DO
 7 1:6   163               ATTRIBUTES^.NDESCRIPTOR[J]:=0;
 7 1:5   188             ATTRIBUTES^.DESCRIPTOR:='                              ';
 7 1:5   223             PUT(ATTRIBUTES);
 7 1:5   231             IF(EOF(ATTRIBUTES))THEN
 7 1:6   241               BEGIN
 7 1:7   241                 WRITELN('OUT OF DISK SPACE');
 7 1:7   278                 ANYKEY;
 7 1:7   281                 BRANCHOUT;
 7 1:7   284                 SETCHAIN('GREETING');
 7 1:7   298                 EXIT(PROGRAM);
 7 1:6   302                 END;
 7 1:4   302             END;
 7 1:3   312           CLOSE(ATTRIBUTES,LOCK);
 7 1:3   321           OPENATTRIBUTESFILE;
 7 1:3   323           EXIT(OPENATTRIBUTESFILE);
 7 1:2   327       END;
 7 1:1   327     CLOSE(ATTRIBUTES);
 7 1:0   336     END;
 7 1:0   356
```

OPENATTRIBUTESFILE creates attributes file if it does not already exist on the disk.

```
118  8    1:D     1 (*$P*)SEGMENT PROCEDURE OPENMEASURESFILE;
119  8    1:0     0   BEGIN
120  8    1:0     0     (*$I-*)
121  8    1:1     0     RESET(MEASURES,NAMEMEASURES);
122  8    1:1    11     (*$I+*)
123  8    1:1    11     I:=IORESULT;
124  8    1:1    16     IF I<>0 THEN
125  8    1:2    23       BEGIN
126  8    1:3    23         WRITELN('Please bear with me while I create the measures file on the disk');
127  8    1:3   107         REWRITE(MEASURES,NAMEMEASURES);
128  8    1:3   120         FOR I:=1 TO NMEASURES DO
129  8    1:4   136           BEGIN
130  8    1:5   136             SEEK(MEASURES,I);
131  8    1:5   147             MEASURES^.DESCRIPTOR:='                              ';
132  8    1:5   182             PUT(MEASURES);
133  8    1:5   190             IF(EOF(MEASURES))THEN
134  8    1:6   200               BEGIN
135  8    1:7   200                 WRITELN('OUT OF DISK SPACE');
136  8    1:7   237                 ANYKEY;
137  8    1:7   240                 BRANCHOUT;
138  8    1:7   243                 SETCHAIN('GREETING');
139  8    1:7   257                 EXIT(PROGRAM);
140  8    1:6   261                 END;
141  8    1:4   261             END;
142  8    1:3   271         CLOSE(MEASURES,LOCK);
143  8    1:3   280         OPENMEASURESFILE;
144  8    1:3   282         EXIT(OPENMEASURESFILE);
145  8    1:2   286         END;
146  8    1:1   286     CLOSE(MEASURES);
147  8    1:0   295     END;
148  8    1:0   314
```

OPENMEASURESFILE creates measures file if it does not already exist on the disk.

```
 9    1:D     1 (*$P*)SEGMENT PROCEDURE READATTRFILE;
 9    1:0     0   BEGIN
 9    1:0     0   (*$I-*)
 9    1:1     0   RESET(ATTRFILE,NAMEATCORE);
 9    1:1    11   I:=IORESULT;
 9    1:0    16   (*$I+*);
 9    1:1    16   IF I<>0 THEN
 9    1:2    23     BEGIN
 9    1:3    23       REWRITE(ATTRFILE,NAMEATCORE);
 9    1:3    36       FOR I:=1 TO NATTRIBUTES DO
 9    1:4    52         BEGIN
 9    1:5    52           ATTRCORE[I]:=0;
 9    1:5    79           ATTRFILE^:=ATTRCORE[I];
 9    1:5   107           PUT(ATTRFILE);
 9    1:5   115           IF EOF(ATTRFILE) THEN
 9    1:6   125             BEGIN
 9    1:7   125               WRITELN('OUT OF DISK SPACE');
 9    1:6   162               END;
 9    1:4   162           END;
 9    1:3   172         ATTRLAST:=0;
 9    1:3   187         NATTRLAST:=0;
 9    1:3   191         ATTRFILE^:=ATTRLAST;
 9    1:3   207         PUT(ATTRFILE);
 9    1:3   215         CLOSE(ATTRFILE,LOCK);
 9    1:2   224         END
 9    1:1   224     ELSE
 9    1:2   226       BEGIN
 9    1:3   226         FOR I:=1 TO NATTRIBUTES DO
 9    1:4   242           BEGIN
 9    1:5   242             GET(ATTRFILE);
 9    1:5   250             ATTRCORE[I]:=ATTRFILE^;
 9    1:4   278             END;
 9    1:3   288         GET(ATTRFILE);
 9    1:3   296         ATTRLAST:=ATTRFILE^;
 9    1:3   312         NATTRLAST:=TRUNC(ATTRLAST);
 9    1:3   325         CLOSE(ATTRFILE);
 9    1:2   334         END;
 9    1:0   334     END;
 9    1:0   352
```

READATTRFILE loads core with index to attributes file.

```
10    1:D     1 (*$P*)SEGMENT PROCEDURE READMEASFILE;
10    1:0     0   BEGIN
10    1:0     0   (*$I-*)
10    1:1     0   RESET(MEASFILE,NAMEMECORE);
10    1:1    11   I:=IORESULT;
10    1:0    16   (*$I+*);
10    1:1    16   IF I<>0 THEN
10    1:2    23     BEGIN
10    1:3    23       REWRITE(MEASFILE,NAMEMECORE);
10    1:3    36       FOR I:=1 TO NMEASURES DO
10    1:4    52         BEGIN
10    1:5    52           MEASCORE[I]:=0;
10    1:5    79           MEASFILE^:=MEASCORE[I];
10    1:5   107           PUT(MEASFILE);
10    1:5   115           IF EOF(MEASFILE) THEN
10    1:6   125             BEGIN
10    1:7   125               WRITELN('OUT OF DISK SPACE');
10    1:7   162               BRANCHOUT;
10    1:7   165               SETCHAIN('GREETING');
10    1:7   179               EXIT(PROGRAM);
10    1:6   183               END;
10    1:4   183           END;
10    1:3   193         MEASLAST:=0;
10    1:3   208         NMEASLAST:=0;
10    1:3   212         MEASFILE^:=MEASLAST;
10    1:3   228         PUT(MEASFILE);
10    1:3   236         CLOSE(MEASFILE,LOCK);
10    1:2   245         END
10    1:1   245     ELSE
10    1:2   247       BEGIN
10    1:3   247         FOR I:=1 TO NMEASURES DO
10    1:4   263           BEGIN
10    1:5   263             GET(MEASFILE);
10    1:5   271             MEASCORE[I]:=MEASFILE^;
10    1:4   299             END;
10    1:3   309         GET(MEASFILE);
10    1:3   317         MEASLAST:=MEASFILE^;
10    1:3   333         NMEASLAST:=TRUNC(MEASLAST);
10    1:3   346         CLOSE(MEASFILE);
10    1:2   355         END;
10    1:0   355     END;
10    1:0   376
```

READMEASFILE loads core with index to measures file.

```
11    1:D     1 (*$P*)SEGMENT PROCEDURE OPENDATAFILE;
11    1:0     0   BEGIN
11    1:0     0     (*$I-*)
11    1:1     0     RESET(DATANODE,DATANAME);
11    1:1    11     (*$I+*)
11    1:1    11     I:=IORESULT;
11    1:1    16     IF I<>0 THEN
11    1:2    23       BEGIN
11    1:3    23         WRITELN('CREATE DATABASE BEFORE ATTRIBUTES AND MEASURES!');
11    1:3    90         BRANCHOUT;
11    1:3    93         SETCHAIN('GREETING');
11    1:3   107         EXIT(PROGRAM);
11    1:2   111         END;
11    1:0   111     END;
11    1:0   124
```

OPENDATAFILE checks to be sure performance item file exists.

```
12    1:D      1 (*$P*)SEGMENT PROCEDURE DEFINEASPECTS;
12    1:0      0   BEGIN
12    1:1      0     ASPECT[1]:='Potentialities';
12    1:1     30     ASPECT[2]:='Processes';
12    1:1     55     ASPECT[3]:='Products';
12    1:1     79     ASPECT[4]:='Environment';
12    1:1    106     ASPECT[5]:='Constraints';
12    1:0    133     END;
12    1:0    146
```

DEFINEASPECTS tells the computer the labels for the aspects file.

```
254  13    1:D     1 (*$P*)SEGMENT PROCEDURE READCOREFILE;
255  13    1:0     0   BEGIN
256  13    1:0     0   (*$I-*)
257  13    1:1     0   RESET(COREFILE,CORENAME);
258  13    1:1    11   I:=IORESULT;
259  13    1:1    16   (*$I+*)
260  13    1:1    16   IF I<>0 THEN
261  13    1:2    23     BEGIN
262  13    1:3    23       WRITELN('COREFILE DOES NOT EXIST');
263  13    1:3    66       ANYKEY;
264  13    1:3    69       BRANCHOUT;
265  13    1:3    72       SETCHAIN('GREETING');
266  13    1:3    86       EXIT(PROGRAM);
267  13    1:2    90       END
268  13    1:1    90     ELSE
269  13    1:2    92       FOR I:=1 TO 300 DO
270  13    1:3   108         BEGIN
271  13    1:4   108           GET(COREFILE);
272  13    1:4   116           CORE[I]:=COREFILE^;
273  13    1:3   144           END;
274  13    1:1   154       GET(COREFILE);
275  13    1:1   162       CORELAST:=COREFILE^;
276  13    1:1   178       NCORELAST:=TRUNC(CORELAST);
277  13    1:1   191       CLOSE(COREFILE)
278  13    1:0   200     END;
279  13    1:0   214
```

READCOREFILE reads index to performance items into core.

```
14    1:D     1 (*$P*)SEGMENT PROCEDURE SORTCOREFILE;
14    1:0     0   BEGIN
14    1:1     0     FOR I:=1 TO 300 DO
14    1:2    16       CORE2[I]:=I;
14    1:1    45     I:=2;
14    1:1    49     REPEAT
14    1:2    49       IF CORE[I]<CORE[I-1] THEN
14    1:3    94         BEGIN
14    1:4    94           TEMP:=CORE[I];
14    1:4   122           CORE[I]:=CORE[I-1];
14    1:4   164           CORE[I-1]:=TEMP;
14    1:4   194           TEMP2:=CORE2[I];
14    1:4   213           CORE2[I]:=CORE2[I-1];
14    1:4   247           CORE2[I-1]:=TEMP2;
14    1:4   268           IF I>2 THEN
14    1:5   275             I:=I-1;
14    1:3   283           END
14    1:2   283         ELSE
14    1:3   285           I:=I+1;
14    1:1   293       UNTIL I>NCORELAST;
14    1:0   302     END;
14    1:0   320
14    1:0   320 (*$I #5:UTILITY.TEXT*)
14    1:0   320
```

SORTCOREFILE prepares an away CORE2 which lists the location of each performance item in numeric order.

```
1   4:D     1 (*$P*)PROCEDURE ANYKEY;
1   4:0     0   BEGIN
1   4:1     0     WRITELN(' ');
1   4:1    18     WRITELN('*** Please press any key to continue ***');
1   4:1    78     (*$R-*)
1   4:1    78     READ(ANS);
1   4:1    89     (*$R+*)
1   4:0    89     END;
1   4:0   102
```

ANYKEY displays "Please Press any Key to Continue" then it awaits a Keypress before returning control to the calling procedure.

```
1    5:D     1 (*$P*)PROCEDURE HELPER;
1    5:0     0   BEGIN
1    5:1     0     WRITELN('For help please refer to your APM MANUAL.');
1    5:0    61     END;
1    5:0    74
```

HELPER; due to core limitations, it was not possible to implement the full HELP facility. Thus, this HELPER merely displays the message.

```
1    4:D     1 (*$P*)PROCEDURE ANYKEY;
1    4:0     0   BEGIN
1    4:1     0     WRITELN(' ');
1    4:1    18     WRITELN('*** Please press any key to continue ***');
1    4:1    78     (*$R-*)
1    4:1    78     READ(ANS);
1    4:1    89     (*$R+*)
1    4:0    89     END;
1    4:0   102
```

ANYKEY displays "Please Press any Key to Continue" then it awaits a Keypress before returning control to the calling procedure.

```
1    5:D     1 (*$P*)PROCEDURE HELPER;
1    5:0     0   BEGIN
1    5:1     0     WRITELN('For help please refer to your APM MANUAL.');
1    5:0    61     END;
1    5:0    74
```

HELPER; due to core limitations, it was not possible to implement the full HELP facility. Thus, this HELPER merely displays the message.

```
1    6:D      1 (*$P*)PROCEDURE KEYN;
1    6:D      1   VAR
1    6:D      1     ANSWER: STRING[40];
1    6:D     22     II: ARRAY[1..4] OF INTEGER;
1    6:D     26     OK:BOOLEAN;
1    6:D     27      IIO:INTEGER;
1    6:D     28
1    6:0      0   BEGIN
1    6:0      0     (*$R-*)
1    6:1      0     REPEAT
1    6:2      0       REPEAT
1    6:3      0         ANSWER:='                          ';
1    6:3     27         OK:=TRUE;
1    6:3     30         READLN(ANSWER);
1    6:3     49         IF LENGTH(ANSWER)=0 THEN
1    6:4     57           WRITELN('Please enter the integer again');
1    6:2    107         UNTIL LENGTH(ANSWER)<>0;
1    6:2    115       IF (ANSWER[1]='H') OR (ANSWER[1]='h') THEN
1    6:3    130         HELPER;
1    6:2    132       FOR I:=1 TO 4 DO
1    6:3    147         BEGIN
1    6:4    147           II[I]:=ORD(ANSWER[I])-48;
1    6:4    165           IF (II[I]<0) OR (II[I]>9) THEN
1    6:5    192             BEGIN
1    6:6    192               IF (I=1) OR (II[I]<>(ORD(' ')-48)) THEN
1    6:7    214                 BEGIN
1    6:8    214                   OK:=FALSE;
1    6:8    217                   WRITELN('PLEASE RESPOND WITH A POSITIVE INTEGER');
1    6:7    275                   END;
1    6:5    275               END;
1    6:3    275           END;
1    6:1    285       UNTIL OK=TRUE;
1    6:1    292     IIO:=II[1];
1    6:1    302     FOR I:=2 TO 4 DO
1    6:2    317       BEGIN
1    6:3    317         IF (II[I]>=0) AND (II[I]<=9) THEN
1    6:4    344           IIO:=IIO*10+II[I];
1    6:2    361         END;
1    6:2    371     (*$R+*)
1    6:1    371     I:=IIO;
1    6:0    376     END;
1    6:0    398
```

KEYN reads a 3 or 4 digit response from the keyboard and places it into I. If an H or an h are typed in, it places a 999 in I and calls the HELP routine. If more than 4 characters are typed, only 4 characters are read. The rest are ignored. If the character(s) are not positive intergers, KEYN will display an appropriate warning and wait for a response.

```
 1    7:D     1 (*$P*)PROCEDURE KEY;
 1    7:D     1   VAR
 1    7:D     1     II2:INTEGER;
 1    7:0     0   BEGIN
 1    7:0     0     (*$R-*)
 1    7:1     0     ANSWER:='                  ';
 1    7:1    27     REPEAT
 1    7:2    27       READLN(ANSWER);
 1    7:2    47       ANS:=ANSWER[1];
 1    7:2    55       IF (ANS<>'Y') AND (ANS<>'N') AND (ANS<>'H') AND (ANS<>'y') and
 1    7:2    78         (ANS<>'n') AND (ANS<>'h') AND (ORD(ANS)<>27)THEN
 1    7:3    98           WRITELN('PLEASE RESPOND YES OR NO!');
 1    7:2   143       IF (ORD(ANS)>90) THEN
 1    7:3   150         BEGIN
 1    7:4   150           II2:=ORD(ANS)-32;
 1    7:4   157           ANS:=CHR(II2);
 1    7:3   161           END;
 1    7:1   161       UNTIL (ANS='Y') OR (ANS='N') OR (ANS='H') OR (ORD(ANS)=27);
 1    7:1   186       (*$R+*)
 1    7:1   186       IF ANS='H' THEN
 1    7:2   193         HELPER;
 1    7:0   195       END;
 1    7:0   210
```

KEY reads a letter response from the keyboard. If response is 1) y or Y, it places a Y in ANS and returns to calling procedure; 2) n or N, it places an N in ANS and returns to calling procedure; 3) h or H, it calls the HELP routine, places an H in ANS and returns to calling program; or 4) any other key--it displays PLEASE RESPOND YES OR NO and awaits a Y, N, H, y, n or h response. NOTE: Only the first character/line is processed. The rest is ignored.

```
1    8:D     1 (*$P*)PROCEDURE PREPKEY(HLP:INTEGER;MSG:STRING);
1    8:0     0   BEGIN
1    8:1     0     HELP:=HLP;
1    8:1     9     REPEAT
1    8:2     9       WRITE(MSG);
1    8:2    20       KEY;
1    8:1    22       UNTIL (ANS='Y') OR (ANS='N') OR (ORD(ANS)=27);
1    8:0    41     END;
1    8:0    56
```

PREPKEY displays a message then calls KEY to read a letter response from the keyboard. If a response is not Y, y, N, n, Yes or No, it redisplays the message and, once again, waits for a response.

```
1   9:D     1 (*$P*)PROCEDURE INLINE;
1   9:D     1   VAR
1   9:D     1     LONGLINE:STRING[125];
1   9:D    64     LINEOK:BOOLEAN;
1   9:D    65
1   9:0     0   BEGIN
1   9:1     0     REPEAT
1   9:2     0       READLN(LONGLINE);
1   9:2    19       LINEOK:=TRUE;
1   9:2    22       M:=LENGTH(LONGLINE);
1   9:2    29       IF M>68 THEN
1   9:3    36         BEGIN
1   9:4    36           WRITELN('**WARNING LINE CONTAINS OVER 68 CHARACTERS**');
1   9:4   100           WRITELN(' ');
1   9:4   118           PREPKEY(39,'DO YOU WISH TO TRUNCATE TO 68 CHARACTERS? ');
1   9:4   166           IF ANS='N' THEN
1   9:5   173             BEGIN
1   9:6   173               LINEOK:=FALSE;
1   9:6   176               WRITELN('PLEASE TYPE LINE AGAIN: ');
1   9:5   220               END
1   9:4   220             ELSE
1   9:5   222               M:=68;
1   9:3   226           END;
1   9:1   226       UNTIL LINEOK;
1   9:1   230     INLINECALL:=INLINECALL+1;
1   9:1   238     IF INLINECALL>25 THEN
1   9:2   245       BEGIN
1   9:3   245         WRITELN('WARNING--You have typed in over 25 new attributes and/or',
1   9:3   313         chr(13),'  measures--the limit for the demonstration.  Please select',
1   9:3   394         chr(13),'  a different analytic procedure before entering more data',
1   9:3   474         chr(13),'  --or risk losing everything you have done today!');
1   9:3   554         ANYKEY;
1   9:2   556         END;
1   9:1   556     LINER:=COPY(LONGLINE,1,M);
1   9:0   574     END;
1   9:0   592
```

INLINE accepts up to 80 characters of text. If more than 80 characters are specified, it asks if it ought to ignore additional characters. If told to, it does. Otherwise, it allows analyst to re-enter the line.

```
1   10:D     1 (*$P*)PROCEDURE SHOWALINE;
1   10:0     0   BEGIN
1   10:1     0     NLENGTH:=LENGTH(LINE);
1   10:1     8     IF NLENGTH<2 THEN
1   10:2    15       EXIT(SHOWALINE);
1   10:1    19     WHILE (LINE[NLENGTH]=' ') AND (NLENGTH>1) DO
1   10:2    37       NLENGTH:=NLENGTH-1;
1   10:1    47     IF NLENGTH<2 THEN
1   10:2    54       EXIT(SHOWALINE);
1   10:1    58     IF NLENGTH<=LLENGTH THEN
1   10:2    67       BEGIN
1   10:3    67         WRITE(LINE);
1   10:3    79         EXIT(SHOWALINE);
1   10:2    83         END;
1   10:1    83     L:=LLENGTH;
1   10:1    89     WHILE (LINE[L]<>' ') AND (L>1) DO
1   10:2   107       L:=L-1;
1   10:1   117     L:=L-1;
1   10:1   125     IF L>0 THEN
1   10:2   132       BEGIN
1   10:3   132         REGLINE:=COPY(LINE,1,L);
1   10:3   151         WRITELN(REGLINE);
1   10:2   171         END;
1   10:1   171     L:=L+2;
1   10:1   179     NLENGTH:=NLENGTH-L+1;
1   10:1   191     IF NLENGTH<1 THEN
1   10:2   198       EXIT(SHOWALINE);
1   10:1   202     REGLINE:=COPY(LINE,L,NLENGTH);
1   10:1   223     FOR I:=1 TO INDENT DO
1   10:2   239       WRITE(' ');
1   10:1   259     WRITE(REGLINE);
1   10:1   271     PC:=PC+1;
1   10:0   279     END;
1   10:0   298
```

SHOWALINE displays text on the screen. If, by chance, the text is longer than the amount of space available on the current line, the display continues onto a second line.

```
1    2:D     1 (*$P*)PROCEDURE BRANCHIN;
1    2:0     0   BEGIN
1    2:0     0     (*$I-*)
1    2:1     0     RESET(PASSNODE,'PASSTHRU');
1    2:1    18     I:=IORESULT;
1    2:1    23     (*$I+*)
1    2:1    23     IF I<>0 THEN
1    2:2    30       BEGIN
1    2:3    30         WRITELN('PASSTHRU FILE DOES NOT EXIST');
1    2:3    78         WRITELN('  ******FATAL ERROR******');
1    2:3   123         WRITELN('              ',I);
1    2:3   167         ANYKEY;
1    2:3   169         SETCHAIN('PGM1');
1    2:3   179         EXIT(PROGRAM);
1    2:2   183         END;
1    2:1   183     GET(PASSNODE);
1    2:1   190     CURSYS:=PASSNODE^.CURSYS;
1    2:1   198     CURSP:=PASSNODE^.CURSP;
1    2:1   206     CURSUB:=PASSNODE^.CURSUB;
1    2:1   214     PAC:=PASSNODE^.PAC;
1    2:1   220     NCURSYS:=PASSNODE^.NCURSYS;
1    2:1   227     NCURSP:=PASSNODE^.NCURSP;
1    2:1   234     NCURSUB:=PASSNODE^.NCURSUB;
1    2:1   241     NPAC:=PASSNODE^.NPAC;
1    2:1   248     CLOSE(PASSNODE);
1    2:0   256     END;
1    2:0   270
```

BRANCHIN gets information from the PASSTHRU file for use by this program.

```
1   3:D    1   (*$P*)PROCEDURE BRANCHOUT;
1   3:0    0     BEGIN
1   3:1    0       REWRITE(PASSNODE,'PASSTHRU');
1   3:1   20       PASSNODE^.FLAG1:=1;
1   3:1   26       PUT(PASSNODE);
1   3:1   33       CLOSE(PASSNODE,LOCK);
1   3:0   41       END;
1   3:0   54
1   3:0   54
1   3:0   54
1   3:0   54 (*$I #5:UTILITY.TEXT*)
1   3:0   54
```

BRANCHOUT loads the PASSTHRU file with appropriate data for use by called programs.

```
1    11:D      1 (*$P*)PROCEDURE CLOSEATTRFILE;
1    11:0      0   BEGIN
1    11:1      0     RESET(ATTRFILE,NAMEATCORE);
1    11:1     13     FOR I:=1 TO NATTRIBUTES DO
1    11:2     29       BEGIN
1    11:3     29         ATTRFILE^:=ATTRCORE[I];
1    11:3     57         PUT(ATTRFILE);
1    11:2     65         END;
1    11:1     75     ATTRLAST:=NATTRLAST;
1    11:1     92     ATTRFILE^:=ATTRLAST;
1    11:1    108     PUT(ATTRFILE);
1    11:1    116     CLOSE(ATTRFILE);
1    11:0    125     END;
1    11:0    140
```

CLOSEATTRFILE saves index to attributes file on the disk.

```
1    12:D      1 (*$P*)PROCEDURE CLOSEMEASFILE;
1    12:0      0   BEGIN
1    12:1      0     RESET(MEASFILE,NAMEMECORE);
1    12:1     13     FOR I:=1 TO NMEASURES DO
1    12:2     29       BEGIN
1    12:3     29         MEASFILE^:=MEASCORE[I];
1    12:3     57         PUT(MEASFILE);
1    12:2     65         END;
1    12:1     75     MEASLAST:=NMEASLAST;
1    12:1     92     MEASFILE^:=MEASLAST;
1    12:1    108     PUT(MEASFILE);
1    12:1    116     CLOSE(MEASFILE);
1    12:0    125     END;
1    12:0    140
```

CLOSEMEASFILE saves index to the measures file on the disk.

```
1    13:D      1 (*$P*)PROCEDURE SETUPSCREEN;
1    13:0      0   BEGIN
1    13:1      0     I:=TRUNC(CORE[NODE] DIV 1000000);
1    13:1     54     PAC:=ASPECT[I];
1    13:1     72     NPAC:=I;
1    13:1     78     TEMP:=CORE[NODE] DIV 100;
1    13:1    115     TEMP2:=TRUNC(CORE[NODE] DIV 10000);
1    13:1    151     FOR J:=1 TO NCORELAST DO
1    13:2    167       BEGIN
1    13:3    167         IF(TEMP2=CORE[J] DIV 10000) AND (CORE[J] DIV 10000*10000=CORE[J]) THEN
1    13:4    274           BEGIN
1    13:5    274             SEEK(DATANODE,CORE2[J]);
1    13:5    298             GET(DATANODE);
1    13:5    306             XOBJECTIVE:=DATANODE^.TAXA;
1    13:5    316             NOBJECTIVE:=DATANODE^.NTAXA[2];
1    13:4    331             END;
1    13:3    331         IF(TEMP=CORE[J] DIV 100) AND(CORE[J]DIV 100 * 100=CORE[J]) THEN
1    13:4    431           BEGIN
1    13:5    431             SEEK(DATANODE,CORE2[J]);
1    13:5    455             GET(DATANODE);
1    13:5    463             XFUNPUR:=DATANODE^.TAXA;
1    13:5    473             NFUNPUR:=DATANODE^.NTAXA[3];
1    13:4    488             END;
1    13:3    488         IF CORE[NODE]=CORE[J] THEN
1    13:4    531           BEGIN
1    13:5    531             SEEK(DATANODE,CORE2[J]);
1    13:5    555             GET(DATANODE);
1    13:5    563             XCHARAC:=DATANODE^.TAXA;
1    13:5    573             NCHARAC:=DATANODE^.NTAXA[4];
1    13:4    588             END;
1    13:2    588         END;
1    13:0    598     END;
1    13:0    614
```

SETUPSCREEN sets up header for the top of each page [or screen] with appropriate information. The header contains the system class, system, subsystem, aspect, objectives, functional purpose and characteristics information for the attributes and/or measures on the display.

```
1   14:D     1 (*$P*)PROCEDURE TOPPAGE;
1   14:0     0   BEGIN
1   14:1     0     PAGE(PRNT);
1   14:1    10     M:=LENGTH(CURSYS);
1   14:1    18     IF M>16 THEN
1   14:2    25       M:=16;
1   14:1    29     LINE:=COPY(CURSYS,1,M);
1   14:1    48     WRITE(PRNT,'*',LINE,' Systems');
1   14:1    90     N:=16-LENGTH(CURSYS);
1   14:1   100     FOR L:=1 TO N DO
1   14:2   116       WRITE(PRNT,' ');
1   14:1   136     M:=LENGTH(CURSP);
1   14:1   144     IF M>16 THEN
1   14:2   151       M:=16;
1   14:1   155     LINE:=COPY(CURSP,1,M);
1   14:1   174     WRITE(PRNT,'*',LINE);
1   14:1   196     N:=16-LENGTH(CURSP);
1   14:1   206     FOR L:=1 TO N DO
1   14:2   222       WRITE(PRNT,' ');
1   14:1   242     M:=LENGTH(CURSUB);
1   14:1   250     IF M>16 THEN
1   14:2   257       M:=16;
1   14:1   261     LINE:=COPY(CURSUB,1,M);
1   14:1   280     WRITE(PRNT,'*',LINE);
1   14:1   302     N:=16-LENGTH(CURSUB);
1   14:1   312     FOR L:=1 TO N DO
1   14:2   328       WRITE(PRNT,' ');
1   14:1   348     WRITELN(PRNT,'*',FAC);
1   14:1   378     IF NPRINT>=1 THEN
1   14:2   385       WRITELN(PRNT,'Objective:[',NOBJECTIVE,']',XOBJECTIVE);
1   14:1   450     IF NPRINT>=2 THEN
1   14:2   457       WRITELN(PRNT,'Fctl Prps:[',NFUNPUR,']',XFUNPUR);
1   14:1   522     IF NPRINT=3 THEN
1   14:2   529       WRITELN(PRNT,'Charstics:[',NCHARAC,']',XCHARAC);
1   14:1   594     WRITELN(PRNT,' ');
1   14:0   612     END;
1   14:0   630
```

TOPPAGE prints appropriate header information at the top of each page.

```
1   15:D     1   (*$P*)PROCEDURE TOPSCREEN;
1   15:0     0     BEGIN
1   15:1     0       PAGE(OUTPUT);
1   15:1    10       M:=LENGTH(CURSYS);
1   15:1    18       IF M>16 THEN
1   15:2    25         M:=16;
1   15:1    29       LINE:=COPY(CURSYS,1,M);
1   15:1    48       WRITE('*',LINE,' Systems');
1   15:1    90       GOTOXY(26,0);
1   15:1    95       M:=LENGTH(CURSP);
1   15:1   103       IF M>16 THEN
1   15:2   110         M:=16;
1   15:1   114       LINE:=COPY(CURSP,1,M);
1   15:1   133       WRITE('*',LINE);
1   15:1   155       GOTOXY(44,0);
1   15:1   160       M:=LENGTH(CURSUB);
1   15:1   168       IF M>16 THEN
1   15:2   175         M:=16;
1   15:1   179       LINE:=COPY(CURSUB,1,M);
1   15:1   198       WRITELN('*',LINE);
1   15:1   228       GOTOXY(62,0);
1   15:1   233       WRITELN('*',PAC);
1   15:1   263       M:=LENGTH(XOBJECTIVE);
1   15:1   271       IF M>67 THEN M:=67;
1   15:1   282       LINE:=COPY(XOBJECTIVE,1,M);
1   15:1   301       IF NSCREEN>=1 THEN
1   15:2   308         WRITELN('Objective[',NOBJECTIVE,']:',LINE);
1   15:1   376       M:=LENGTH(XFUNPUR);
1   15:1   384       IF M>67 THEN M:=67;
1   15:1   395       LINE:=COPY(XFUNPUR,1,M);
1   15:1   414       IF NSCREEN>=2 THEN
1   15:2   421         WRITELN('Fctl Prps[',NFUNPUR,']:',LINE);
1   15:1   489       M:=LENGTH(XCHARAC);
1   15:1   497       IF M>67 THEN M:=67;
1   15:1   508       LINE:=COPY(XCHARAC,1,M);
1   15:1   527       IF NSCREEN=3 THEN
1   15:2   534         WRITELN('Chrctstcs[',NCHARAC,']:',LINE);
1   15:1   602       WRITELN(' ');
1   15:0   620       END;
1   15:0   632
```

TOPSCREEN displays appropriate header information at the top of each screen.

```
1   16:D     1 (*$P*)PROCEDURE PRNTATTRLINE;
1   16:0     0   BEGIN
1   16:1     0     RESET(ATTRIBUTES,NAMEATTRIBUTES);
1   16:1    13     SEEK(ATTRIBUTES,NCURATTRIBUTE);
1   16:1    24     GET(ATTRIBUTES);
1   16:1    32     K:=ATTRIBUTES^.NDESCRIPTOR[5];
1   16:1    47     LINE:=ATTRIBUTES^.DESCRIPTOR;
1   16:1    57     WRITELN(PRNT,LINE,'[',K,']');
1   16:1   109     CLOSE(ATTRIBUTES);
1   16:0   118     END;
1   16:0   130
```

PRNTATTRLINE prints one attribute when called by PRNTTOP.

```
  1   17:D     1 (*$P*)PROCEDURE ATTRLINEDISPLAY;
  1   17:0     0   BEGIN
  1   17:1     0     RESET(ATTRIBUTES,NAMEATTRIBUTES);
  1   17:1    13     SEEK(ATTRIBUTES,NCURATTRIBUTE);
  1   17:1    24     GET(ATTRIBUTES);
  1   17:1    32     K:=ATTRIBUTES^.NDESCRIPTOR[5];
  1   17:1    47     LINE:=ATTRIBUTES^.DESCRIPTOR;
  1   17:1    57     M:=LENGTH(LINE);
  1   17:1    65     IF M>67 THEN
  1   17:2    72       M:=67;
  1   17:1    76     LINE:=COPY(LINE,1,M);
  1   17:1    95     WRITELN(LINE,'[',K,']');
  1   17:1   147     CLOSE(ATTRIBUTES);
  1   17:0   156     END;
  1   17:0   168
```

ATTRLINEDISPLAY adds an attribute to the header for a measurement item display.

```
1   18:D     1 (*$P*)PROCEDURE PRINTONEATTRIBUTE;
1   18:0     0   BEGIN
1   18:1     0     RESET(ATTRIBUTES,NAMEATTRIBUTES);
1   18:1    13     SEEK(ATTRIBUTES,NCURATTRIBUTE);
1   18:1    24     GET(ATTRIBUTES);
1   18:1    32     WRITE(PRNT,NATTR,'.  [');
1   18:1    60     FOR J:=1 TO 5 DO
1   18:2    74       BEGIN
1   18:3    74         K:=ATTRIBUTES^.NDESCRIPTOR[J];
1   18:3    91         WRITE(PRNT,K,'.');
1   18:2   113         END;
1   18:1   123     LINE:=ATTRIBUTES^.DESCRIPTOR;
1   18:1   133     WRITELN(PRNT,']',LINE);
1   18:1   163     CLOSE(ATTRIBUTES);
1   18:0   172     END;
1   18:0   186
```

PRINTONEATTRIBUTE prints one attribute in the body of the attribute display.

```
  1   19:D     1 (*$P*)PROCEDURE ONEATTRIBUTEDISPLAY;
  1   19:0     0   BEGIN
  1   19:1     0     RESET(ATTRIBUTES,NAMEATTRIBUTES);
  1   19:1    13     SEEK(ATTRIBUTES,NCURATTRIBUTE);
  1   19:1    24     GET(ATTRIBUTES);
  1   19:1    32     WRITE(NATTR,'.  [');
  1   19:1    60     FOR J:=1 TO 5 DO
  1   19:2    74       BEGIN
  1   19:3    74         K:=ATTRIBUTES^.NDESCRIPTOR[J];
  1   19:3    91         WRITE(K,'.');
  1   19:2   113         END;
  1   19:1   123     LINE:=ATTRIBUTES^.DESCRIPTOR;
  1   19:1   133     LLENGTH:=60;
  1   19:1   137     WRITE(']');
  1   19:1   147     INDENT:=16;
  1   19:1   151     SHOWALINE;
  1   19:1   153     WRITELN(' ');
  1   19:1   171     CLOSE(ATTRIBUTES);
  1   19:0   180     END;
  1   19:0   194
```

ONEATTRIBUTEDISPLAY displays one attribute in the body of the attribute display.

```
1   20:D     1 (*$P*)PROCEDURE PRINTTHEATTRIBUTES;
1   20:0     0   BEGIN
1   20:1     0     NATTR:=0;
1   20:1     4     OK:=FALSE;
1   20:1     8     WRITELN(PRNT,'Measurable Attributes--To evaluate effectiveness in meeting this ',
1   20:1    85                 chr(13),'  characteristic, the following system attributes can be
                               measured:');
1   20:1   182     FOR NCURATTRIBUTE:=1 TO NATTRLAST DO
1   20:2   198       BEGIN
1   20:3   198         TEMPX:=ATTRCORE[NCURATTRIBUTE] DIV 100;
1   20:3   235         TEMP:=TEMPX;
1   20:3   251         IF TEMP=CORE[NODE] THEN
1   20:4   282           BEGIN
1   20:5   282             OK:=TRUE;
1   20:5   286             NATTR:=NATTR+1;
1   20:5   294             PRINTONEATTRIBUTE;
1   20:4   296             END;
1   20:2   296         END;
1   20:1   306     IF OK=FALSE THEN
1   20:2   314       WRITELN(PRNT,'    ...none');
1   20:0   344     END;
1   20:0   358
```

PRINTTHEATTRIBUTES prints the body of the attribute display on the printer.

```
1   21:D    1 {$$P$}PROCEDURE SHOWATTRIBUTES;
1   21:0    0   BEGIN
1   21:1    0     NATTR:=0;
1   21:1    4     FOR I:=1 TO 20 DO
1   21:2   18       BEGIN
1   21:3   18         ATTRINDEX[I]:=0;
1   21:3   33         MEASINDEX[I]:=0;
1   21:2   48         END;
1   21:1   58     GOTOXY(0,4);
1   21:1   63     WRITE(CHR(11));
1   21:1   73     OK:=FALSE;
1   21:1   77     DISPCOUNT:=0;
1   21:1   81     WRITELN('Measurable Attributes--To evaluate effectiveness in meeting this ',
1   21:1  158             chr(13),'  characteristic, the following attributes can be measured:');
1   21:1  255     FOR NCURATTRIBUTE:=1 TO NATTRLAST DO
1   21:2  271       BEGIN
1   21:3  271         TEMPX:=ATTRCORE[NCURATTRIBUTE] DIV 100;
1   21:3  308         TEMP:=TEMPX;
1   21:3  324         IF TEMP=CORE[NODE] THEN
1   21:4  355           BEGIN
1   21:5  355             IF DISPCOUNT >=10 THEN
1   21:6  362               BEGIN
1   21:7  362                 DISPCOUNT:=0;
1   21:7  366                 ANYKEY;
1   21:7  368                 GOTOXY(0,6);
1   21:7  373                 WRITE(CHR(11));
1   21:6  383                 END;
1   21:5  383             OK:=TRUE;
1   21:5  387             NATTR:=NATTR+1;
1   21:5  395             ATTRINDEX[NATTR]:=NCURATTRIBUTE;
1   21:5  412             PC:=0;
1   21:5  416             ONEATTRIBUTEDISPLAY;
1   21:5  418             IF PC=1 THEN
1   21:6  425               DISPCOUNT:=DISPCOUNT+1;
1   21:5  433             PC:=0;
1   21:5  437             DISPCOUNT:=DISPCOUNT+1;
1   21:4  445             END;
1   21:2  445         END;
1   21:1  455     IF OK=FALSE THEN
1   21:2  463       WRITELN('   ...none');
1   21:0  493     END;
1   21:0  512
```

SHOWATTRIBUTES displays the body of the attribute display on the screen.

```
1   22:D     1 (*$P*)PROCEDURE REWORDATTRIBUTES;
1   22:0     0   BEGIN
1   22:1     0     IF OK THEN
1   22:2     5       BEGIN
1   22:3     5         REPEAT
1   22:4     5           GOTOXY(0,15);
1   22:4    10           WRITE(CHR(11));
1   22:4    20           WRITE('Which one (type 0 if done) ?');
1   22:4    60           KEYN;
1   22:4    62           IF (I<0) OR (I>NATTR) THEN
1   22:5    77             BEGIN
1   22:6    77               WRITELN('Please type an integer between 0 and ',NATTR,'.');
1   22:6   156               ANYKEY;
1   22:5   158               END;
1   22:3   158           UNTIL (I>=0) AND (I<=NATTR);
1   22:3   173         IF I<>0 THEN
1   22:4   180           BEGIN
1   22:5   180             I:=ATTRINDEX[I];
1   22:5   197             WRITELN('Please type the new attribute descriptor: ');
1   22:5   259             WRITE('.............');
1   22:5   284             INLINE;
1   22:5   286             RESET(ATTRIBUTES,NAMEATTRIBUTES);
1   22:5   299             SEEK(ATTRIBUTES,I);
1   22:5   310             GET(ATTRIBUTES);
1   22:5   318             IF LENGTH(LINER)<69 THEN
1   22:6   327               ATTRIBUTES^.DESCRIPTOR:=LINER
1   22:5   332                 ELSE
1   22:6   339                   ATTRIBUTES^.DESCRIPTOR:=COPY(LINER,1,68);
1   22:5   358             SEEK(ATTRIBUTES,I);
1   22:5   369             PUT(ATTRIBUTES);
1   22:5   377             CLOSE(ATTRIBUTES);
1   22:4   386             END;
1   22:2   386         END
1   22:1   386         ELSE
1   22:2   388           BEGIN
1   22:3   388             WRITELN('There are no attributes for this performance item');
1   22:3   457             ANYKEY;
1   22:2   459             END;
1   22:0   459       END;
1   22:0   478
1   22:0   478
```

REWORDATTRIBUTES asks which attribute to reword. Then it asks the analyst to reword the attribute.

```
 1   23:D     1 (*$P*)PROCEDURE DELETEATTRIBUTES;
 1   23:0     0   BEGIN
 1   23:1     0     IF OK THEN
 1   23:2     5       BEGIN
 1   23:3     5         REPEAT
 1   23:4     5           GOTOXY(0,15);
 1   23:4    10           WRITE(CHR(11));
 1   23:4    20           WRITE('Which one (type 0 if done) ?');
 1   23:4    60           KEYN;
 1   23:4    62           IF (I<0) OR (I>NATTR) THEN
 1   23:5    77             BEGIN
 1   23:6    77               WRITELN('Please type an integer between 0 and ',NATTR,'.');
 1   23:6   156               ANYKEY;
 1   23:5   158               END;
 1   23:3   158           UNTIL (I>=0) AND (I<=NATTR);
 1   23:3   173         IF I<>0 THEN
 1   23:4   180           BEGIN
 1   23:5   180             I:=ATTRINDEX[I];
 1   23:5   197             RESET(ATTRIBUTES,NAMEATTRIBUTES);
 1   23:5   210             SEEK(ATTRIBUTES,I);
 1   23:5   221             GET(ATTRIBUTES);
 1   23:5   229             FOR J:=1 TO 6 DO
 1   23:6   243               ATTRIBUTES^.NDESCRIPTOR[J]:=0;
 1   23:5   268             ATTRIBUTES^.DESCRIPTOR:='                    ';
 1   23:5   298             SEEK(ATTRIBUTES,I);
 1   23:5   309             PUT(ATTRIBUTES);
 1   23:5   317             CLOSE(ATTRIBUTES);
 1   23:5   326             ATTRCORE[I]:=0;
 1   23:4   353             END;
 1   23:2   353         END
 1   23:1   353         ELSE
 1   23:2   355           BEGIN
 1   23:3   355             WRITELN('There are no attributes for this performance item');
 1   23:3   424             ANYKEY;
 1   23:2   426             END;
 1   23:0   426     END;
 1   23:0   446
```

DELETEATTRIBUTES asks analyst which attribute to delete. Then it deletes the attribute.

```
1   24:D     1 (*$P*)PROCEDURE ADDATTRIBUTES;
1   24:0     0   BEGIN
1   24:1     0     IF NATTRLAST>=200 THEN
1   24:2     9       BEGIN
1   24:3     9         WRITELN('DATASET CONTAINS 200 ATTRIBUTE LIMIT');
1   24:3    65         ANYKEY;
1   24:2    67         END;
1   24:1    67     FOR J:=1 TO 20 DO
1   24:2    81       SCRATCH[J]:=J;
1   24:1   108     FOR J:=1 TO NATTRLAST DO
1   24:2   124       IF CORE[NODE]=(ATTRCORE[J] DIV 100) THEN
1   24:3   176         BEGIN
1   24:4   176           K:=TRUNC(ATTRCORE[J]-ATTRCORE[J] DIV 100 * 100);
1   24:4   241           IF K<>0 THEN
1   24:5   248             SCRATCH[K]:=0;
1   24:3   263           END;
1   24:3   273     (*$I-*)
1   24:1   273     RESET(ATTRIBUTES,NAMEATTRIBUTES);
1   24:1   284     (*$I+*)
1   24:1   284     FOR J:=1 TO 4 DO
1   24:2   298       ATTRIBUTES^.NDESCRIPTOR[J]:=DATANODE^.NTAXA[J];
1   24:1   336     ATTRIBUTES^.NDESCRIPTOR[6]:=0;
1   24:1   349     GOTOXY(0,15);
1   24:1   354     WRITE(CHR(11));
1   24:1   364     WRITELN('Please type the new attribute descriptor:');
1   24:1   425     WRITE('.............');
1   24:1   450     INLINE;
1   24:1   452     IF LINER='' THEN
1   24:2   462       BEGIN
1   24:3   462       CLOSE(ATTRIBUTES);
1   24:3   471       EXIT(ADDATTRIBUTES);
1   24:2   475       END;
1   24:1   475     NATTRLAST:=NATTRLAST+1;
1   24:1   483     SEEK(ATTRIBUTES,NATTRLAST);
1   24:1   494     FOR J:=20 DOWNTO 1 DO
1   24:2   508       IF SCRATCH[J]<>0 THEN
1   24:3   526         BEGIN
1   24:4   526           ATTRIBUTES^.NDESCRIPTOR[5]:=J;
1   24:4   541           K:=J;
1   24:3   547           END;
```

ADDATTRIBUTES asks the analyst to type in a new attribute, then it adds the attribute to the attribute list.

```
881   1   24:1   557      IF LENGTH(LINER)<69 THEN
882   1   24:2   566        ATTRIBUTES^.DESCRIPTOR:=LINER
883   1   24:1   571        ELSE
884   1   24:2   578          ATTRIBUTES^.DESCRIPTOR:=COPY(LINER,1,68);
885   1   24:1   597      PUT(ATTRIBUTES);
886   1   24:1   605      TEMPX:=CORE[NODE]*100+K;
887   1   24:1   653      SCRATCH[K]:=0;
888   1   24:1   668      ATTRCORE[NATTRLAST]:=TEMPX;
889   1   24:1   696      CLOSE(ATTRIBUTES);
890   1   24:1   705      OK:=TRUE;
891   1   24:0   709      END;
892   1   24:0   732
892   1   24:0   732 (*$I #5:MEASATTR2.TEXT *)
```

See previous page for program description.

```
1   25:D     1 (*$P*)PROCEDURE PRINTAMEASURE;
1   25:0     0   BEGIN
1   25:1     0     RESET(MEASURES,NAMEMEASURES);
1   25:1    13     SEEK(MEASURES,NCURMEASURE);
1   25:1    24     GET(MEASURES);
1   25:1    32     WRITE(PRNT,NMEAS,'.  [');
1   25:1    60     FOR J:=1 TO 6 DO
1   25:2    74       BEGIN
1   25:3    74         K:=MEASURES^.NDESCRIPTOR[J];
1   25:3    91         WRITE(PRNT,K,'.');
1   25:2   113         END;
1   25:1   123     LINE:=MEASURES^.DESCRIPTOR;
1   25:1   133     WRITELN(PRNT,']',LINE);
1   25:1   163     CLOSE(MEASURES);
1   25:0   172     END;
1   25:0   186
```

PRINTAMEASURE prints one measure in the body of the measure display.

```
1   26:D     1 (*$P*)PROCEDURE ONEMEASUREDISPLAY;
1   26:0     0   BEGIN
1   26:1     0     RESET(MEASURES,NAMEMEASURES);
1   26:1    13     SEEK(MEASURES,NCURMEASURE);
1   26:1    24     GET(MEASURES);
1   26:1    32     WRITE(NMEAS,'.  [');
1   26:1    60     FOR J:=1 TO 6 DO
1   26:2    74       BEGIN
1   26:3    74         K:=MEASURES^.NDESCRIPTOR[J];
1   26:3    91         WRITE(K,'.');
1   26:2   113         END;
1   26:1   123     LINE:=MEASURES^.DESCRIPTOR;
1   26:1   133     LLENGTH:=60;
1   26:1   137     WRITE(']');
1   26:1   147     INDENT:=18;
1   26:1   151     SHOWALINE;
1   26:1   153     WRITELN(' ');
1   26:1   171     CLOSE(MEASURES);
1   26:0   180     END;
1   26:0   194
```

ONEMEASUREDISPLAY displays one measure in the body of the measure display.

```
1    27:D      1 (*$P*)PROCEDURE PRNTTHEMEASURES;
1    27:0      0   BEGIN
1    27:1      0     NMEAS:=0;
1    27:1      4     OK:=FALSE;
1    27:1      8     WRITELN(PRNT,'Measures--This system attribute can be analyzed by comparing',
1    27:1     80                chr(13),'  the following parameters with established criteria:');
1    27:1    163     FOR NCURMEASURE:=1 TO NMEASLAST DO
1    27:2    179       BEGIN
1    27:3    179         TEMPX:=MEASCORE[NCURMEASURE] DIV 100;
1    27:3    216         IF TEMPX=ATTRCORE[NCURATTRIBUTE] THEN
1    27:4    247           BEGIN
1    27:5    247             OK:=TRUE;
1    27:5    251             NMEAS:=NMEAS+1;
1    27:5    259             PRINTAMEASURE;
1    27:4    261             END;
1    27:2    261         END;
1    27:1    271     IF OK=FALSE THEN
1    27:2    279       WRITELN(PRNT,'   ...none');
1    27:0    309     END;
1    27:0    324
```

PRNTTHEMEASURES prints the body of the measure display.

```
1   28:D     1 (*$P*)PROCEDURE SHOWMEASURES;
1   28:0     0   BEGIN
1   28:1     0     NMEAS:=0;
1   28:1     4     FOR I:=1 TO 20 DO
1   28:2    18       MEASINDEX[I]:=0;
1   28:1    43     GOTOXY(0,7);
1   28:1    48     WRITE(CHR(11));
1   28:1    58     OK:=FALSE;
1   28:1    62     DISPMCOUNT:=1;
1   28:1    66     WRITELN('Measures--This system attribute can be analyzed by comparing',
1   28:1   138             chr(13),'  the following parameters with established criteria:');
1   28:1   221     FOR NCURMEASURE:=1 TO NMEASLAST DO
1   28:2   237       BEGIN
1   28:3   237         TEMPX:=MEASCORE[NCURMEASURE] DIV 100;
1   28:3   274         IF TEMPX=ATTRCORE[NCURATTRIBUTE] THEN
1   28:4   305           BEGIN
1   28:5   305             IF DISPMCOUNT >6 THEN
1   28:6   312               BEGIN
1   28:7   312                 DISPMCOUNT:=1;
1   28:7   316                 ANYKEY;
1   28:7   318                 GOTOXY(0,9);
1   28:7   323                 WRITE(CHR(11));
1   28:6   333                 END;
1   28:5   333             OK:=TRUE;
1   28:5   337             NMEAS:=NMEAS+1;
1   28:5   345             MEASINDEX[NMEAS]:=NCURMEASURE;
1   28:5   362             PC:=0;
1   28:5   366             ONEMEASUREDISPLAY;
1   28:5   368             IF PC=1 THEN
1   28:6   375               DISPMCOUNT:=DISPMCOUNT+1;
1   28:5   383             PC:=0;
1   28:5   387             DISPMCOUNT:=DISPMCOUNT+1;
1   28:4   395             END;
1   28:2   395         END;
1   28:1   405     IF OK=FALSE THEN
1   28:2   413       WRITELN('    ...none');
1   28:0   443     END;
1   28:0   462
```

SHOWMEASURES displays the body of the measure display on the screen.

```
1   29:D    1 (*$P*)PROCEDURE REWORDMEASURES;
1   29:0    0   BEGIN
1   29:1    0     REPEAT
1   29:2    0       GOTOXY(0,15);
1   29:2    5       WRITE(CHR(11));
1   29:2   15       WRITE('Which one (type 0 if done) ?');
1   29:2   55       KEYN;
1   29:2   57       IF (I<0) OR (I>NMEAS) THEN
1   29:3   72           BEGIN
1   29:4   72             WRITELN('please type an integer between 0 and ',NMEAS,'.');
1   29:4  151             ANYKEY;
1   29:3  153             END;
1   29:1  153       UNTIL (I>=0) AND (I<=NMEAS);
1   29:1  168     IF I<>0 THEN
1   29:2  175       BEGIN
1   29:3  175         I:=MEASINDEX[I];
1   29:3  192         WRITELN('Please type the new measure descriptor: ');
1   29:3  252         WRITE('.............');
1   29:3  277         INLINE;
1   29:3  279         RESET(MEASURES,NAMEMEASURES);
1   29:3  292         SEEK(MEASURES,I);
1   29:3  303         GET(MEASURES);
1   29:3  311         IF LENGTH (LINER)<69 THEN
1   29:4  320           MEASURES^.DESCRIPTOR:=LINER
1   29:3  325           ELSE
1   29:4  332           MEASURES^.DESCRIPTOR:=COPY(LINER,1,68);
1   29:3  351         SEEK(MEASURES,I);
1   29:3  362         PUT(MEASURES);
1   29:3  370         CLOSE(MEASURES);
1   29:2  379         END
1   29:1  379       ELSE
1   29:2  381         BEGIN
1   29:3  381           WRITELN('There are no measures for this attribute');
1   29:3  441           ANYKEY;
1   29:2  443           END;
1   29:0  443     END;
1   29:0  460
1   29:0  460
```

REWORDMEASURES asks analyst which measure to reword. Then it asks him/her to reword the measure.

```
 1   30:D     1 (*$P*)PROCEDURE DELETEMEASURES;
 1   30:0     0   BEGIN
 1   30:1     0     REPEAT
 1   30:2     0       GOTOXY(0,15);
 1   30:2     5       WRITE(CHR(11));
 1   30:2    15       WRITE('Which one (type 0 if done) ?');
 1   30:2    55       KEYN;
 1   30:2    57       IF (I<0) OR (I>NMEAS) THEN
 1   30:3    72         BEGIN
 1   30:4    72           WRITELN('Please type an integer between 0 and ',NMEAS,'.');
 1   30:4   151           ANYKEY;
 1   30:3   153           END;
 1   30:1   153       UNTIL (I>=0) OR (I<=NMEAS);
 1   30:1   168     IF I<>0 THEN
 1   30:2   175       BEGIN
 1   30:3   175         I:=MEASINDEX[I];
 1   30:3   192         RESET(MEASURES,NAMEMEASURES);
 1   30:3   205         SEEK(MEASURES,I);
 1   30:3   216         GET(MEASURES);
 1   30:3   224         FOR J:=1 TO 6 DO
 1   30:4   238           MEASURES^.NDESCRIPTOR[J]:=0;
 1   30:3   263         MEASURES^.DESCRIPTOR:='                    ';
 1   30:3   293         SEEK(MEASURES,I);
 1   30:3   304         PUT(MEASURES);
 1   30:3   312         CLOSE(MEASURES);
 1   30:3   321         MEASCORE[I]:=0;
 1   30:2   348         END
 1   30:1   348       ELSE
 1   30:2   350         BEGIN
 1   30:3   350           WRITELN('There are no measures for this attribute');
 1   30:3   410           ANYKEY;
 1   30:2   412           END;
 1   30:0   412     END;
 1   30:0   430
```

DELETEMEASURES asks the analyst which measure to delete. Then it deletes it.

```
1   31:D     1 (*$P*)PROCEDURE ADDMEASURES;
1   31:0     0   BEGIN
1   31:1     0     IF NMEASLAST>=400 THEN
1   31:2     9       BEGIN
1   31:3     9         WRITELN('DATA SET CONTAINS LIMIT OF 400 MEASURES');
1   31:3    68         ANYKEY;
1   31:3    70         EXIT(ADDMEASURES);
1   31:2    74         END;
1   31:1    74     FOR J:=1 TO 20 DO
1   31:2    88       SCRATCH[J]:=J;
1   31:1   115     FOR J:=1 TO NMEASLAST DO
1   31:2   131       IF ATTRCORE[NCURATTRIBUTE]=(MEASCORE[J] DIV 100) THEN
1   31:3   183         BEGIN
1   31:4   183           K:=TRUNC(MEASCORE[J]-MEASCORE[J] DIV 100 * 100);
1   31:4   248           IF K<>0 THEN
1   31:5   255             SCRATCH[K]:=0;
1   31:3   270           END;
1   31:3   280     (*$I-*)
1   31:1   280     RESET(MEASURES,NAMEMEASURES);
1   31:1   291     (*$I+*)
1   31:1   291     FOR J:=1 TO 5 DO
1   31:2   305       MEASURES^.NDESCRIPTOR[J]:=ATTRIBUTES^.NDESCRIPTOR[J];
1   31:1   343     GOTOXY(0,15);
1   31:1   348     WRITE(CHR(11));
1   31:1   358     WRITELN('Please type the new measure descriptor (68 characters available:');
1   31:1   442     WRITE('.............');
1   31:1   467     INLINE;
1   31:1   469     IF LINER='' THEN
1   31:2   479       BEGIN
1   31:3   479       CLOSE(MEASURES);
1   31:3   488       EXIT(ADDMEASURES);
1   31:2   492       END;
1   31:1   492     NMEASLAST:=NMEASLAST+1;
1   31:1   500     SEEK(MEASURES,NMEASLAST);
1   31:1   511     FOR J:=20 DOWNTO 1 DO
1   31:2   525       IF SCRATCH[J]<>0 THEN
1   31:3   543         BEGIN
1   31:4   543           MEASURES^.NDESCRIPTOR[6]:=J;
1   31:4   558           K:=J;
1   31:3   564           END;
```

ADDMEASURES asks analyst to type in the new measures.

```
1099  1  31:1  574     if length(liner)<69 then
1100  1  31:2  583       MEASURES^.DESCRIPTOR:=LINER
1101  1  31:1  588       else
1102  1  31:2  595         measures^.descriptor:=copy(liner,1,68);
1103  1  31:1  614     PUT(MEASURES);
1104  1  31:1  622     TEMPX:=ATTRCORE[NCURATTRIBUTE]*100+K;
1105  1  31:1  670     SCRATCH[K]:=0;
1106  1  31:1  685     MEASCORE[NMEASLAST]:=TEMPX;
1107  1  31:1  713     CLOSE(MEASURES);
1108  1  31:0  722     END;
1109  1  31:0  744
```

See previous page for program description.

```
1   32:D     1 (*$P*)PROCEDURE PRINTMEASURES;
1   32:0     0   BEGIN
1   32:1     0     REWRITE(PRNT,'PRINTER:');
1   32:1    21     TOPPAGE;
1   32:1    23     WRITELN(PRNT,'Measurable Attributes--To evaluate effectiveness in meeting this',
1   32:1    99                chr(13),'  characteristic, the following system attributes can be
                              measured: ');
1   32:1   196     WRITE('     ');
1   32:1   212     PRNTATTRLINE;
1   32:1   214     PRNTTHEMEASURES;
1   32:1   216     CLOSE(PRNT);
1   32:0   225     END;
1   32:0   238
```

PRINTMEASURES controls printout of entire page of measures.

```
1 33:D   1 (*$P*)PROCEDURE EXAMINEMEASURES;
1 33:0   0   BEGIN
1 33:1   0     TOPSCREEN;
1 33:1   2     GOTOXY(0,4);
1 33:1   7     WRITE(CHR(11));
1 33:1  17     WRITELN('Measurable Attributes--To evaluate effectiveness in meeting this',
1 33:1  93          chr(13),' characteristic, the following system attributes can be measured: ');
1 33:1 190     WRITE('    ');
1 33:1 206     ATTRLINEDISPLAY;
1 33:1 208       SHOWMEASURES;
1 33:1 210       GOTOXY(0,15);
1 33:1 215       WRITE(CHR(11));
1 33:1 225       WRITE('You may perform the following procedures:',chr(13),
1 33:1 288 '  1. Add new measures                2. Reword a measure',chr(13),
1 33:1 363 '  3. Remove a measure                4. Print these measures',chr(13),
1 33:1 442 '  5. Return to Attributes Level   ',chr(13),
1 33:1 498 'Please select one: ');
1 33:1 529       REPEAT
1 33:2 529         KEYN;
1 33:2 531         IF (I<1) OR (I>5) THEN
1 33:3 544           WRITELN('Please type an integer between 1 and 5');
1 33:1 602         UNTIL (I>=1) AND (I<=5);
1 33:1 615       CASE I OF
1 33:1 620         1:ADDMEASURES;
1 33:1 624         2:REWORDMEASURES;
1 33:1 628         3:DELETEMEASURES;
1 33:1 632         4:PRINTMEASURES;
1 33:1 636         5:EXIT(EXAMINEMEASURES);
1 33:1 642         END;
1 33:1 660       EXAMINEMEASURES;
1 33:0 662       END;
1 33:0 676
1 33:0 676 (*$I #5:MEASATTR2.TEXT *)
1 33:0 676
1 33:0 676 (*$I #5:MEASATTR3.TEXT *)
```

EXAMINEMEASURES controls production of entire display for analyzing measures.

```
1   34:D     1 (*$P*)PROCEDURE ONEPERFITEMDISPLAY;
1   34:0     0   BEGIN
1   34:1     0     SEEK(DATANODE,CORE2[NODE]);
1   34:1    24     GET(DATANODE);
1   34:1    32     K:=DATANODE^.NTAXA[M];
1   34:1    49     LINE:=DATANODE^.TAXA;
1   34:1    59     LLENGTH:=72;
1   34:1    63     IF K<>0 THEN
1   34:2    70       BEGIN
1   34:3    70         WRITE('  ',K,'. ');
1   34:3   110         INDENT:=6;
1   34:3   114         SHOWALINE;
1   34:3   116         WRITELN(' ');
1   34:2   134         END;
1   34:1   134     IF (K=0) AND (M=3) THEN
1   34:2   147       WRITELN('  ',K,'. ','Process at the Objectives level');
1   34:1   238     IF (K=0) AND (M=4) THEN
1   34:2   251       WRITELN('  ',K,'. ','Process at the Functional Purposes level');
1   34:0   351     END;
1   34:0   364
```

ONEPERFITEMDISPLAY displays one performance iten in the body of the display used to select which performance item ought to be processed next.

```
1   35:D     1 (*$P*)PROCEDURE SHOWPERFITEMS;
1   35:0     0   BEGIN
1   35:1     0     OK:=FALSE;
1   35:1     4     DISPCOUNT:=0;
1   35:1     8     IF M=2 THEN TEMPL1:=1000000;
1   35:1    50     IF M=2 THEN TEMPL3:=10000;
1   35:1    74     IF M=3 THEN TEMPL1:=10000;
1   35:1    98     IF M=3 THEN TEMPL3:=100;
1   35:1   120     IF M=4 THEN TEMPL1:=100;
1   35:1   142     IF M=4 THEN TEMPL3:=1;
1   35:1   164     TEMPL2:=TEMP DIV TEMPL1;
1   35:1   190     FOR NODE:=1 TO NCORELAST DO
1   35:2   206       BEGIN
1   35:3   206         IF (TEMPL2=CORE[NODE] DIV TEMPL1) AND
1   35:3   245          (CORE[NODE] DIV TEMPL3 * TEMPL3 = CORE[NODE]) THEN
1   35:4   309           BEGIN
1   35:5   309             OK:=TRUE;
1   35:5   313             ONEPERFITEMDISPLAY;
1   35:5   315             DISPCOUNT:=DISPCOUNT+1;
1   35:4   323             END;
1   35:3   323         IF (DISPCOUNT DIV 10 * 10=DISPCOUNT) AND (DISPCOUNT<>0) THEN
1   35:4   342           BEGIN
1   35:5   342             DISPCOUNT:=0;
1   35:5   346             ANYKEY;
1   35:5   348             GOTOXY(0,5);
1   35:5   353             WRITE(CHR(11));
1   35:4   363             END;
1   35:2   363         END;
1   35:1   373       IF OK=FALSE THEN
1   35:2   381         WRITELN('    ...none');
1   35:0   411       END;
1   35:0   428
```

SHOWPERFITEMS controls production of the body of displays of performance items.

```
1208 1 36:D     1 (*$P*)PROCEDURE CHANGECHARACTERISTICS;
1209 1 36:0     0   BEGIN
1210 1 36:1     0     TOPSCREEN;
1211 1 36:1     2     GOTOXY(0,4);
1212 1 36:1     7     WRITE(CHR(11));
1213 1 36:1    17     N:=4;
1214 1 36:1    21     WRITELN('The following Characteristics are available for the Functional Purpose');
1215 1 36:1   111     SHOWPERFITEMS;
1216 1 36:1   113     WRITE('Please select one: ');
1217 1 36:1   144     KEYN;
1218 1 36:1   146     NCHARAC:=I;
1219 1 36:1   152     TEMPL4:=I;
1220 1 36:1   169     TEMP:=TEMP+TEMPL4;
1221 1 36:1   195     FOR I:=1 TO 300 DO
1222 1 36:2   211       IF TEMP=CORE[I] THEN
1223 1 36:3   242         BEGIN
1224 1 36:4   242           SEEK(DATANODE,CORE2[I]);
1225 1 36:4   266           GET(DATANODE);
1226 1 36:4   274           XCHARAC:=DATANODE^.TAXA;
1227 1 36:3   284           END;
1228 1 36:1   294       NSCREEN:=3;
1229 1 36:1   298       NPRINT:=3;
1230 1 36:0   302     END;
1231 1 36:0   316
```

CHANGECHARACTERISTICS governs producing the list of characteristics when analyst is selecting a different performance item.

```
1 37:D    (*$P*)PROCEDURE CHANGEFUNCTIONALPURPOSES;
1 37:0   0  BEGIN
1 37:1   0    TOPSCREEN;
1 37:1   2    GOTOXY(0,4);
1 37:1   7    WRITE(CHR(11));
1 37:1  17    M:=3;
1 37:1  21    WRITELN('The following Functional Purposes are available for the objective selected: ');
1 37:1 117    SHOWPERFITEMS;
1 37:1 119    WRITE('Please select one: ');
1 37:1 150    KEYN;
1 37:1 152    TEMPL4:=I;
1 37:1 169    NFUNPUR:=I;
1 37:1 175    TEMP:=TEMP+TEMPL4*100;
1 37:1 210    FOR I:=1 TO 300 DO
1 37:2 226      IF TEMP=CORE[I] THEN
1 37:3 257        BEGIN
1 37:4 257          SEEK(DATANODE,CORE2[I]);
1 37:4 281          GET(DATANODE);
1 37:4 289          XFUNPUR:=DATANODE^.TAXA;
1 37:3 299          END;
1 37:1 309    IF NFUNPUR<>0 THEN
1 37:2 316      BEGIN
1 37:3 316        NSCREEN:=2;
1 37:3 320        NPRINT:=2;
1 37:3 324        CHANGECHARACTERISTCS;
1 37:2 326        END;
1 37:0 326    END;
1 37:0 340
```

CHANGEFUNCTIONALPURPOSES governs producing a list of functional purposes when analyst is selecting a different performance item.

```
1   38:D     1 (*$P*)PROCEDURE CHANGENODE;
1   38:0     0   BEGIN
1   38:1     0     REPEAT
1   38:2     0       NSCREEN:=0;
1   38:2     4       NPRINT:=0;
1   38:2     8       PAC:='              ';
1   38:2    28       TOPSCREEN;
1   38:2    30       GOTOXY(0,4);
1   38:2    35       WRITE(CHR(11));
1   38:2    45       WRITE('The following aspects are part of the APM:',chr(13),
1   38:2   109             '  1. Potentialities',chr(13),
1   38:2   150             '  2. Processes',chr(13),
1   38:2   186             '  3. Products',chr(13),
1   38:2   221             '  4. Environment',chr(13),
1   38:2   259             '  5. Constraints',chr(13),
1   38:2   297             'Please select one: ');
1   38:2   328       KEYN;
1   38:2   330       PAC:=ASPECT[I];
1   38:2   348       NPAC:=I;
1   38:2   354       TEMPL4:=I;
1   38:2   371       TEMP:=TEMPL4*1000000;
1   38:2   416       GOTOXY(0,0);
1   38:2   421       WRITE(CHR(11));
1   38:2   431       TOPSCREEN;
1   38:2   433       M:=2;
1   38:2   437       WRITELN('The following Objectives are available for the aspect selected: ');
1   38:2   521       SHOWPERFITEMS;
1   38:2   523       WRITE('Please select one: ');
1   38:2   554       KEYN;
1   38:2   556       NOBJECTIVE:=I;
1   38:2   562       TEMPL4:=I;
1   38:2   579       TEMP:=TEMP+TEMPL4*10000;
1   38:2   616       FOR I:=1 TO 300 DO
1   38:3   632         IF TEMP=CORE[I] THEN
1   38:4   663           BEGIN
1   38:5   663             SEEK(DATANODE,CORE2[I]);
1   38:5   687             GET(DATANODE);
1   38:5   695             XOBJECTIVE:=DATANODE^.TAXA;
1   38:4   705             END;
1   38:2   715       IF NOBJECTIVE<>0 THEN
```

CHANGENODE is the master routine to specify a different performance item for analysis.

```
1    38:3    722          BEGIN
1    38:4    722            NSCREEN:=1;
1    38:4    726            NPRINT:=1;
1    38:4    730            CHANGEFUNCTIONALPURPOSES;
1    38:3    732            END;
1    38:2    732        NODE:=0;
1    38:2    736        FOR I:=1 TO 300 DO
1    38:3    752          IF CORE[I]=TEMP THEN
1    38:4    783            BEGIN
1    38:5    783              NODE:=I-1;
1    38:4    791              END;
1    38:2    801        IF NODE=0 THEN
1    38:3    808          BEGIN
1    38:4    808            WRITELN('Performance item # ',temp,' does not exist!');
1    38:4    895            ANYKEY;
1    38:3    897            END;
1    38:1    897        UNTIL NODE<>0;
1    38:0    904        END;
1    38:0    922
```

See previous page for program description.

```
1   39:D     1 (*$P*)PROCEDURE PRINTATTRIBUTES;
1   39:0     0   BEGIN
1   39:1     0     REWRITE(PRNT,'PRINTER:');
1   39:1    21     TOPPAGE;
1   39:1    23     PRINTTHEATTRIBUTES;
1   39:1    25     CLOSE(PRNT);
1   39:0    34     END;
1   39:0    46
```

PRINTATTRIBUTES controls printout of entire page of attributes.

```
1   40:D     1 (*$P*)PROCEDURE EXAMINEATTRIBUTES; FORWARD;
1   40:D     1
```

These procedures are presented later on in this program.

```
1   41:D     1 (*$P*)PROCEDURE PREPEXATTR;
1   41:0     0   BEGIN
1   41:1     0     REPEAT
1   41:2     0       NODE:=NODE+1;
1   41:2     8       IF CORE[NODE] DIV 10000 * 10000=CORE[NODE] THEN
1   41:3    73         BEGIN
1   41:4    73           NSCREEN:=1;
1   41:4    77           NPRINT:=1;
1   41:4    81           SEEK(DATANODE,CORE2[NODE]);
1   41:4   105           GET(DATANODE);
1   41:4   113           XOBJECTIVE:=DATANODE^.TAXA;
1   41:4   123           NOBJECTIVE:=DATANODE^.NTAXA[2];
1   41:3   138           END;
1   41:2   138       IF (CORE[NODE] DIV 100 * 100 = CORE[NODE])
1   41:2   197        AND(CORE[NODE] DIV 1000 * 1000<>CORE[NODE]) THEN
1   41:3   263         BEGIN
1   41:4   263           NSCREEN:=2;
1   41:4   267           NPRINT:=2;
1   41:4   271           SEEK(DATANODE,CORE2[NODE]);
1   41:4   295           GET(DATANODE);
1   41:4   303           XFUNPUR:=DATANODE^.TAXA;
1   41:4   313           NFUNPUR:=DATANODE^.NTAXA[3];
1   41:3   328           END;
1   41:2   328       IF CORE[NODE] DIV 100 * 100 <> CORE[NODE] THEN
1   41:3   389         BEGIN
1   41:4   389           NSCREEN:=3;
1   41:4   393           NPRINT:=3;
1   41:4   397           SEEK(DATANODE,CORE2[NODE]);
1   41:4   421           GET(DATANODE);
1   41:4   429           XCHARAC:=DATANODE^.TAXA;
1   41:4   439           NCHARAC:=DATANODE^.NTAXA[4];
1   41:3   454           END;
1   41:2   454       I:=TRUNC(CORE[NODE] DIV 1000000);
1   41:1   508       UNTIL I<>0;
1   41:1   515     PAC:=ASPECT[I];
1   41:1   533     NPAC:=I;
1   41:0   539     END;
1   41:0   554
```

PREPEXATTR sets up header for an attributes analysis display.

```
  1   42:D     1 (*$P*)PROCEDURE PREEXAMINEATTRIBUTES;
  1   42:0     0   BEGIN
  1   42:1     0     NODE:=0;
  1   42:1     4     PREPKEY(109,'Would you like to begin analyzing the first performance item?');
  1   42:1    71     IF ANS='N' THEN
  1   42:2    78       CHANGENODE;
  1   42:1    80     PREPEXATTR;
  1   42:1    82     EXAMINEATTRIBUTES;
  1   42:0    84     END;
  1   42:0    96
```

PREEXAMINEATTRIBUTES prepares computer to process the first performance item.

```
1   43:D     1 (*$P*)PROCEDURE EXMEAS;
1   43:0     0   BEGIN
1   43:1     0     IF OK THEN
1   43:2     5       BEGIN
1   43:3     5         REPEAT
1   43:4     5           WRITE('Which one (type 0 to reconsider) ?');
1   43:4    51           KEYN;
1   43:4    53           IF (I<0) OR (I>NATTR) THEN
1   43:5    68             BEGIN
1   43:6    68               WRITELN('Please type an integer between 0 and ',NATTR,'.');
1   43:6   147               ANYKEY;
1   43:5   149               END;
1   43:3   149           UNTIL (I>=0) AND (I<=NATTR);
1   43:3   164         IF I=0 THEN
1   43:4   171           EXIT(EXMEAS);
1   43:3   175         NCURATTRIBUTE:=ATTRINDEX[I];
1   43:3   192         IF NCURATTRIBUTE>0 THEN
1   43:4   199           EXAMINEMEASURES;
1   43:2   201         END
1   43:1   201       ELSE
1   43:2   203         BEGIN
1   43:3   203           GOTOXY(0,15);
1   43:3   208           WRITE(CHR(11));
1   43:3   218           WRITELN('There are no attributes to analyze');
1   43:3   272           ANYKEY;
1   43:2   274           END;
1   43:0   274     END;
1   43:0   290
```

EXMEAS asks analyst which measure he wishes to analyze.

```
1405 1 40:D   1 (*$P*)PROCEDURE EXAMINEATTRIBUTES;
1406 1 40:0   0   BEGIN
1407 1 40:1   0     REPEAT
1408 1 40:2   0       TOPSCREEN;
1409 1 40:2   2       SHOWATTRIBUTES;
1410 1 40:2   4       GOTOXY(0,15);
1411 1 40:2   9       WRITELN(CHR(11));
1412 1 40:2  27       WRITE(' You may perform any of the following procedures:', chr(13),
1413 1 40:2  98 '  1. Examine measures for an attribute     2. Add new attributes',chr(13),
1414 1 40:2 183 '  3. Reword an attribute                   4. Remove an attribute',chr(13),
1415 1 40:2 269 '  5. Print these attributes                6. Proceed to the NEXT perf item',chr(13),
1416 1 40:2 365 '  7. Proceed to ANOTHER perf item          8. Select a different analytic proc',chr(13),
1417 1 40:2 464 'Please select one: ');
1418 1 40:2 495       REPEAT
1419 1 40:3 495         KEYN;
1420 1 40:3 497         IF (I<1) OR (I>8) THEN
1421 1 40:4 510           WRITELN('Please type an integer between 1 and 8');
1422 1 40:2 568         UNTIL (I>=1) AND (I<=8);
1423 1 40:2 581       CASE I OF
1424 1 40:2 586         1:EXMEAS;
1425 1 40:2 590         2:ADDATTRIBUTES;
1426 1 40:2 594         3:REWORDATTRIBUTES;
1427 1 40:2 598         4:DELETEATTRIBUTES;
1428 1 40:2 602         5:PRINTATTRIBUTES;
1429 1 40:2 606         6:PREPEXATTR;
1430 1 40:2 610         7:BEGIN
1431 1 40:4 610             CHANGENODE;
1432 1 40:4 612             PREPEXATTR;
1433 1 40:3 614             END;
1434 1 40:2 616         8:EXIT(EXAMINEATTRIBUTES);
1435 1 40:2 622         END;
1436 1 40:1 646         UNTIL NODE>=NCORELAST;
1437 1 40:1 655       WRITELN('All performance items have been processed');
1438 1 40:1 716       PREPKEY(99,'Do you wish to review any items?');
1439 1 40:1 754       IF ANS='N' THEN
1440 1 40:2 761         EXIT(EXAMINEATTRIBUTES);
1441 1 40:1 765       CHANGENODE;
1442 1 40:1 767       EXAMINEATTRIBUTES;
1443 1 40:0 769 END;
1444 1 40:0 786
1445 1 40:0 786 (*$I #5:MEASATTR3.TEXT *)
1446 1 40:0 786
```

EXAMINEATTRIBUTES governs setting up an entire display for examining attributes.

```
1 1:0   0(*$P*)BEGIN
1 1:0   0  (*$N-*)
1 1:1   0  INLINECALL:=0;
1 1:1  93  NMEASURES:=400;
1 1:1  99  NATTRIBUTES:=200;
1 1:1 105  BRANCHIN;
1 1:1 107  DEFINEASPECTS;
1 1:1 110  APMDSK:=CONCAT(COPY(CURSYS,1,2),(COPY(CURSP,1,2)),COPY(CURSUB,1,2),':');
1 1:1 202  NAMEATCORE:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),(COPY(CURSP,1,4)),(COPY(CURSUB,1,4)),'AC');
1 1:1 305  NAMEATTRIBUTES:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),(COPY(CURSP,1,4)),(COPY(CURSUB,
           1,4)),'AT');
1 1:1 408  NAMEMECORE:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),(COPY(CURSP,1,4)),(COPY(CURSUB,1,4)),'MC');
1 1:1 511  NAMEMEASURES:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),(COPY(CURSP,1,4)),(COPY(CURSUB,1,
           4)),'ME');
1 1:1 614  CORENAME:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),(COPY(CURSP,1,4)),(COPY(CURSUB,1,4)),'CO');
1 1:1 717  DATANAME:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),(COPY(CURSP,1,4)),(COPY(CURSUB,1,4)),'FI');
1 1:1 820  OPENDATAFILE;
1 1:1 823  OPENATTRIBUTESFILE;
1 1:1 826  OPENMEASURESFILE;
1 1:1 829  READATTRFILE;
1 1:1 832  READMEASFILE;
1 1:1 835  READCOREFILE;
1 1:1 838  SORTCOREFILE;
1 1:1 841  PREEXAMINEATTRIBUTES;
1 1:1 843  CLOSEATTRFILE;
1 1:1 845  CLOSEMEASFILE;
1 1:1 847  BRANCHOUT;
1 1:1 849  SETCHAIN('GREETING');
1 1:0 863  END.
```

MAINROUTINE for specifying attributes and measures.

MEASUREMENT PURPOSE PROGRAM (MEASPURP)

The measurement purpose program allows the analyst to edit measurement purposes (adding, rewording and deleting as appropriate). It also allows the analyst to associate (or disassociate) measurement purposes with characteristics. As characteristics are associated and disassociated, the corresponding objectives and functional purposes are treated in a similar way. Thus, objectives and functional purposes are never associated (or disassociated) directly.

```
 1    1:D    1 (*$L PRINTER: *)
 1    1:D    1 (*$S+*)
 1    1:D    1 (* Program to compose measurement purpose index*)
 1    1:D    1 (* Ronald G. Shapiro    Version 2.0    10/25/82*)
 1    1:D    1 Program Formissue;
 1    1:D    3
28    1:D    3
28    2:D    1   PROCEDURE SETCHAIN(TYTLE:STRING);
28    3:D    1   PROCEDURE SETCVAL(VAL:STRING);
28    4:D    1   PROCEDURE GETCVAL(VAR VAL:STRING);
28    5:D    1   PROCEDURE SWAPON;
28    6:D    1   PROCEDURE SWAPOFF;
28    6:D    1
 1    1:D    1 Uses Chainstuff;
 1    1:D    3
```

These procedures are part of the Apple Computer's CHAINSTUFF library entry. The demonstration package uses only SETCHAIN which causes another program to be activated.

```
1    1:D    3 (*$P*)TYPE
1    1:D    3   PASSFILE =RECORD
1    1:D    3     CURSYS,CURSP,CURSUB,PAC:STRING[80];
1    1:D    3     NCURSYS,NCURSP,NCURSUB,NPAC,FLAG1,FLAG2,FLAG3:INTEGER;
1    1:D    3     END;
1    1:D    3
1    1:D    3   DATABASE=RECORD
1    1:D    3     NTAXA: ARRAY[1..4] OF INTEGER;
1    1:D    3     TAXA: STRING[80];
1    1:D    3     END;
1    1:D    3
1    1:D    3   ISSUEFILE =RECORD
1    1:D    3     NUM:INTEGER;
1    1:D    3     NAME: ARRAY[1..2] OF STRING[80];
1    1:D    3     DATA: ARRAY[1..225] OF INTEGER[8];
1    1:D    3     END;
1    1:D    3
1    1:D    3   FASTFILE=RECORD
1    1:D    3     PRINTIT:PACKED ARRAY[1..300]OF BOOLEAN;
1    1:D    3     END;
1    1:D    3
```

PASSFILE for communication between programs [see GREETING program]. DATABASE contains a basic list of performance items. ISSUEFILE contains measurement purpose names and references to performance items. FASTFILE allows for fast printout of repeated performance items.

```
37   1   1:D     3 (*$P*)VAR
38   1   1:D     3   DATANODE:FILE OF DATABASE;
39   1   1:D   348   COREFILE:FILE OF INTEGER[8];
40   1   1:D   651   PASSNODE:FILE OF PASSFILE;
41   1   1:D  1122   ISSUE:FILE OF ISSUEFILE;
42   1   1:D  2180   FASTISSUE: FILE OF FASTFILE;
43   1   1:D  2499
44   1   1:D  2499   XCHARAC,XFUNPUR,XOBJECTIVE,PAC,CURSYS,CURSP,CURSUB: STRING[80];
45   1   1:D  2786   NCURISSUE,NCHARAC,NFUNPUR,NOBJECTIVE,NPAC,NCURSYS,NCURSP,NCURSUB: INTEGER;
46   1   1:D  2794
47   1   1:D  2794   APMDSK:STRING[8];
48   1   1:D  2799   NAMEFASTISSUE,CORENAME,DATANAME,ISSUENAME: STRING[24];
49   1   1:D  2851
50   1   1:D  2851   CORE: ARRAY[1..300] OF INTEGER[8];
51   1   1:D  3751   CORE2: ARRAY[1..300] OF INTEGER;
52   1   1:D  4051
53   1   1:D  4051   ISSUEDATA: ARRAY[1..225] OF INTEGER[8];
54   1   1:D  4726
55   1   1:D  4726   FLAG: ARRAY[1..300] OF BOOLEAN;
56   1   1:D  5026
57   1   1:D  5026   ASPECT: ARRAY[1..5] OF STRING[14];
58   1   1:D  5066   INVERSEA: ARRAY[1..5] OF INTEGER;
59   1   1:D  5071
60   1   1:D  5071   DISPCOUNT,GOPAGE,COUNT,INVERSE,HELP,NSCREEN,NODE:INTEGER;
61   1   1:D  5078   NCORELAST,NISSUES,NUISSUES,ITEMCOUNT:INTEGER;
62   1   1:D  5082   TEMPL1,TEMPL2,TEMPL3,TEMPL4,TEMPL5,TEMPL6,TEMP,CORELAST: INTEGER[8];
63   1   1:D  5106   I,J,K,L,M,N,NN,INLINECALL,INDENT,NLENGTH,LLENGTH,PC,TEMP2:INTEGER;
64   1   1:D  5119
65   1   1:D  5119   CHARACTERISTIC,NEXTCHARACTERISTIC,LASTCHARACTERISTIC:BOOLEAN;
66   1   1:D  5122   REFERENCED,LONGWAY,DONE,OVER,OK,SKIP:BOOLEAN;
67   1   1:D  5128
68   1   1:D  5128   LINER:STRING[80];
69   1   1:D  5169   LINE:STRING[80];
70   1   1:D  5210   PROCESS:STRING[15];
71   1   1:D  5218
72   1   1:D  5218   ANSWER,REGLINE:STRING[80];
73   1   1:D  5300
74   1   1:D  5300   ANS,ANSHOLD: CHAR;
75   1   1:D  5302
76   1   2:D     1 PROCEDURE EXAMINEISSUES;FORWARD;
77   1   3:D     1 PROCEDURE COMPACTISSUES;FORWARD;
78   1   3:D     1
78   1   3:D     1 (*$I #5:UTILITY.TEXT*)
79   1   3:D     1
```

These strings, arrays and variables are used by this program.

```
1    4:D     1 {$$P$}PROCEDURE ANYKEY;
1    4:0     0   BEGIN
1    4:1     0     WRITELN(' ');
1    4:1    18     WRITELN('$$$ Please press any key to continue $$$');
1    4:1    78     {$$R-$}
1    4:1    78     READ(ANS);
1    4:1    89     {$$R+$}
1    4:0    89     END;
1    4:0   102
```

ANYKEY displays "Please Press any Key to Continue" then it awaits a Keypress before returning control to the calling procedure.

```
1    5:D     1 (*$P*)PROCEDURE HELPER;
1    5:0     0   BEGIN
1    5:1     0     WRITELN('For help please refer to your APM MANUAL.');
1    5:0    61     END;
1    5:0    74
```

HELPER; due to core limitations, it was not possible to implement the full HELP facility. Thus, this HELPER merely displays the message.

```
 1   6:D     1 (*$P*)PROCEDURE KEYN;
 1   6:D     1   VAR
 1   6:D     1     ANSWER: STRING[40];
 1   6:D    22     II: ARRAY[1..4] OF INTEGER;
 1   6:D    26     OK:BOOLEAN;
 1   6:D    27      IIO:INTEGER;
 1   6:D    28
 1   6:0     0   BEGIN
 1   6:0     0     (*$R-*)
 1   6:1     0     REPEAT
 1   6:2     0       REPEAT
 1   6:3     0         ANSWER:='                         ';
 1   6:3    27         OK:=TRUE;
 1   6:3    30         READLN(ANSWER);
 1   6:3    49         IF LENGTH(ANSWER)=0 THEN
 1   6:4    57           WRITELN('Please enter the integer again');
 1   6:2   107         UNTIL LENGTH(ANSWER)<>0;
 1   6:2   115       IF (ANSWER[1]='H') OR (ANSWER[1]='h') THEN
 1   6:3   130         HELPER;
 1   6:2   132       FOR I:=1 TO 4 DO
 1   6:3   147         BEGIN
 1   6:4   147           II[I]:=ORD(ANSWER[I])-48;
 1   6:4   165           IF (II[I]<0) OR (II[I]>9) THEN
 1   6:5   192             BEGIN
 1   6:6   192               IF (I=1) OR (II[I]<>(ORD(' ')-48)) THEN
 1   6:7   214                 BEGIN
 1   6:8   214                   OK:=FALSE;
 1   6:8   217                   WRITELN('PLEASE RESPOND WITH A POSITIVE INTEGER');
 1   6:7   275                   END;
 1   6:5   275               END;
 1   6:3   275           END;
 1   6:1   285       UNTIL OK=TRUE;
 1   6:1   292     IIO:=II[1];
 1   6:1   302     FOR I:=2 TO 4 DO
 1   6:2   317       BEGIN
 1   6:3   317         IF (II[I]>=0) AND (II[I]<=9) THEN
 1   6:4   344           IIO:=IIO*10+II[I];
 1   6:2   361         END;
 1   6:2   371     (*$R+*)
 1   6:1   371     I:=IIO;
 1   6:0   376     END;
 1   6:0   398
```

KEYN reads a 3 or 4 digit response from the keyboard and places it into I. If an H or an h are typed in, it places a 999 in I and calls the HELP routine. If more than 4 characters are typed, only 4 characters are read. The rest are ignored. If the character(s) are not positive intergers, KEYN will display an appropriate warning and wait for a response.

```
1    7:D      1 (*$P*)PROCEDURE KEY;
1    7:D      1   VAR
1    7:D      1     II2:INTEGER;
1    7:0      0   BEGIN
1    7:0      0     (*$R-*)
1    7:1      0     ANSWER:='                         ';
1    7:1     27     REPEAT
1    7:2     27       READLN(ANSWER);
1    7:2     47       ANS:=ANSWER[1];
1    7:2     55       IF (ANS<>'Y') AND (ANS<>'N') AND (ANS<>'H') AND (ANS<>'y') and
1    7:2     78         (ANS<>'n') AND (ANS<>'h') AND (ORD(ANS)<>27)THEN
1    7:3     98           WRITELN('PLEASE RESPOND YES OR NO!');
1    7:2    143       IF (ORD(ANS)>90) THEN
1    7:3    150         BEGIN
1    7:4    150           II2:=ORD(ANS)-32;
1    7:4    157           ANS:=CHR(II2);
1    7:3    161           END;
1    7:1    161       UNTIL (ANS='Y') OR (ANS='N') OR (ANS='H') OR (ORD(ANS)=27);
1    7:1    186       (*$R+*)
1    7:1    186       IF ANS='H' THEN
1    7:2    193         HELPER;
1    7:0    195       END;
1    7:0    210
```

KEY reads a letter response from the keyboard. If response is 1) y or Y, it places a Y in ANS and returns to calling procedure; 2) n or N, it places an N in ANS and returns to calling procedure; 3) h or H, it calls the HELP routine, places an H in ANS and returns to calling program; or 4) any other key--it displays PLEASE RESPOND YES OR NO and awaits a Y, N, H, y, n or h response. NOTE: Only the first character/line is processed. The rest is ignored.

```
1      8:D      1 (*$P*)PROCEDURE PREPKEY(HLP:INTEGER;MSG:STRING);
1      8:0      0   BEGIN
1      8:1      0-    HELP:=HLP;
1      8:1      9     REPEAT
1      8:2      9       WRITE(MSG);
1      8:2     20       KEY;
1      8:1     22       UNTIL (ANS='Y') OR (ANS='N') OR (ORD(ANS)=27);
1      8:0     41     END;
1      8:0     56
```

PREPKEY displays a message then calls KEY to read a letter response from the keyboard. If a response is not Y, y, N, n, Yes or No, it redisplays the message and, once again, waits for a response.

```
  1    9:D     1 (*$P*)PROCEDURE INLINE;
  1    9:D     1   VAR
  1    9:D     1     LONGLINE:STRING[125];
  1    9:D    64     LINEOK:BOOLEAN;
  1    9:D    65
  1    9:0     0   BEGIN
  1    9:1     0     REPEAT
  1    9:2     0       READLN(LONGLINE);
  1    9:2    19       LINEOK:=TRUE;
  1    9:2    22       M:=LENGTH(LONGLINE);
  1    9:2    29       IF M>68 THEN
  1    9:3    36         BEGIN
  1    9:4    36           WRITELN('**WARNING LINE CONTAINS OVER 68 CHARACTERS**');
  1    9:4   100           WRITELN(' ');
  1    9:4   118           PREPKEY(39,'DO YOU WISH TO TRUNCATE TO 68 CHARACTERS? ');
  1    9:4   166           IF ANS='N' THEN
  1    9:5   173             BEGIN
  1    9:6   173               LINEOK:=FALSE;
  1    9:6   176               WRITELN('PLEASE TYPE LINE AGAIN: ');
  1    9:5   220               END
  1    9:4   220             ELSE
  1    9:5   222               M:=68;
  1    9:3   226           END;
  1    9:1   226       UNTIL LINEOK;
  1    9:1   230     INLINECALL:=INLINECALL+1;
  1    9:1   238     IF INLINECALL>25 THEN
  1    9:2   245       BEGIN
  1    9:3   245         WRITELN('WARNING--You have typed in over 25 new attributes and/or',
  1    9:3   313         chr(13),'  measures--the limit for the demonstration.  Please select',
  1    9:3   394         chr(13),'  a different analytic procedure before entering more data',
  1    9:3   474         chr(13),'  --or risk losing everything you have done today!');
  1    9:3   554         ANYKEY;
  1    9:2   556         END;
  1    9:1   556     LINER:=COPY(LONGLINE,1,M);
  1    9:0   574     END;
  1    9:0   592
```

INLINE accepts up to 80 characters of text. If more than 80 characters are specified, it asks if it ought to ignore additional characters. If told to, it does. Otherwise, it allows analyst to re-enter the line.

```
 1   10:D     1 (*$P*)PROCEDURE SHOWALINE;
 1   10:0     0   BEGIN
 1   10:1     0     NLENGTH:=LENGTH(LINE);
 1   10:1     8     WHILE (LINE[NLENGTH]=' ') AND (NLENGTH>1) DO
 1   10:2    26       NLENGTH:=NLENGTH-1;
 1   10:1    36     IF NLENGTH<2 THEN
 1   10:2    43       EXIT(SHOWALINE);
 1   10:1    47     IF NLENGTH<=LLENGTH THEN
 1   10:2    56       BEGIN
 1   10:3    56         WRITE(LINE);
 1   10:3    68         EXIT(SHOWALINE);
 1   10:2    72         END;
 1   10:1    72     L:=LLENGTH;
 1   10:1    78     WHILE (LINE[L]<>' ') AND (L>1) DO
 1   10:2    96       L:=L-1;
 1   10:1   106     L:=L-1;
 1   10:1   114     IF L>0 THEN
 1   10:2   121       BEGIN
 1   10:3   121         REGLINE:=COPY(LINE,1,L);
 1   10:3   140         WRITELN(REGLINE);
 1   10:2   160         END;
 1   10:1   160     L:=L+2;
 1   10:1   168     NLENGTH:=NLENGTH-L+1;
 1   10:1   180     IF NLENGTH<1 THEN
 1   10:2   187       EXIT(SHOWALINE);
 1   10:1   191     REGLINE:=COPY(LINE,L,NLENGTH);
 1   10:1   212     FOR I:=1 TO INDENT DO
 1   10:2   228       WRITE(' ');
 1   10:1   248     WRITE(REGLINE);
 1   10:1   260     PC:=PC+1;
 1   10:0   268     END;
 1   10:0   284
```

SHOWALINE displays text on the screen. If, by chance, the text is longer than the amount of space available on the current line, the display continues onto a second line.

```
236   1   11:D     1 (*$P*)PROCEDURE BRANCHIN;
237   1   11:0     0   BEGIN
238   1   11:0     0     (*$I-*)
239   1   11:1     0     RESET(PASSNODE,'PASSTHRU');
240   1   11:1    19     I:=IORESULT;
241   1   11:1    24     (*$I+*)
242   1   11:1    24     IF I<>0 THEN
243   1   11:2    31       BEGIN
244   1   11:3    31         WRITELN('PASSTHRU FILE DOES NOT EXIST');
245   1   11:3    79         WRITELN('  ******FATAL ERROR******');
246   1   11:3   124         WRITELN('             ',I);
247   1   11:3   168         ANYKEY;
248   1   11:3   170         SETCHAIN('PGM1');
249   1   11:3   180         EXIT(PROGRAM);
250   1   11:2   184         END;
251   1   11:1   184     GET(PASSNODE);
252   1   11:1   192     CURSYS:=PASSNODE^.CURSYS;
253   1   11:1   202     CURSP:=PASSNODE^.CURSP;
254   1   11:1   212     CURSUB:=PASSNODE^.CURSUB;
255   1   11:1   222     PAC:=PASSNODE^.PAC;
256   1   11:1   230     NCURSYS:=PASSNODE^.NCURSYS;
257   1   11:1   239     NCURSP:=PASSNODE^.NCURSP;
258   1   11:1   248     NCURSUB:=PASSNODE^.NCURSUB;
259   1   11:1   257     NPAC:=PASSNODE^.NPAC;
260   1   11:1   266     CLOSE(PASSNODE);
261   1   11:0   275     END;
262   1   11:0   290
```

BRANCHIN gets information from the PASSTHRU file for use by this program.

```
263   1   12:D    1   (*$P*)PROCEDURE BRANCHOUT;
264   1   12:0    0     BEGIN
265   1   12:1    0       REWRITE(PASSNODE,'PASSTHRU');
266   1   12:1   21       PASSNODE^.FLAG1:=1;
267   1   12:1   29       PUT(PASSNODE);
268   1   12:1   37       CLOSE(PASSNODE,LOCK);
269   1   12:0   46       END;
270   1   12:0   58
271   1   12:0   58
272   1   12:0   58
273   1   12:0   58 (*$I #5:UTILITY.TEXT*)
274   1   12:0   58
```

BRANCHOUT loads the PASSTHRU file with appropriate data for use by called programs.

```
1   13:D     1 (*$P*)PROCEDURE DEFINEASPECTS;
1   13:0     0   BEGIN
1   13:1     0     ASPECT[1]:='Potentialities';
1   13:1    30     ASPECT[2]:='Processes';
1   13:1    55     ASPECT[3]:='Products';
1   13:1    79     ASPECT[4]:='Environment';
1   13:1   106     ASPECT[5]:='Constraints';
1   13:0   133     END;
1   13:0   146
```

DEFINEASPECTS assigns names to each aspect.

```
1   14:0     1 (*$P*)PROCEDURE READCOREFILE;
1   14:0     0   BEGIN
1   14:0     0   (*$I-*)
1   14:1     0   RESET(COREFILE,CORENAME);
1   14:1    11   I:=IORESULT;
1   14:1    16   (*$I+*)
1   14:1    16   IF I<>0 THEN
1   14:2    23     BEGIN
1   14:3    23       WRITELN('COREFILE DOES NOT EXIST');
1   14:3    66       WRITELN('  $$$$FATAL ERROR$$$$  ');
1   14:3   109       WRITELN('              ',I);
1   14:3   152       ANYKEY;
1   14:3   154       BRANCHOUT;
1   14:3   156       SETCHAIN('GREETING');
1   14:3   170       EXIT(PROGRAM);
1   14:2   174       END
1   14:1   174     ELSE
1   14:2   176         FOR I:=1 TO 300 DO
1   14:3   192           BEGIN
1   14:4   192             GET(COREFILE);
1   14:4   200             CORE[I]:=COREFILE^;
1   14:3   228             END;
1   14:1   238         GET(COREFILE);
1   14:1   246         CORELAST:=COREFILE^;
1   14:1   262         NCORELAST:=TRUNC(CORELAST);
1   14:1   275         CLOSE(COREFILE)
1   14:0   284       END;
1   14:0   300
```

READCOREFILE reads performance item index file from disk into core.

```
1   15:D     1 (*$P*)PROCEDURE SORTCOREFILE;
1   15:0     0   BEGIN
1   15:1     0     FOR I:=1 TO 300 DO
1   15:2    16       CORE2[I]:=I;
1   15:1    45     I:=2;
1   15:1    49     REPEAT
1   15:2    49       IF CORE[I]<CORE[I-1] THEN
1   15:3    94         BEGIN
1   15:4    94           TEMP:=CORE[I];
1   15:4   122           CORE[I]:=CORE[I-1];
1   15:4   164           CORE[I-1]:=TEMP;
1   15:4   194           TEMP2:=CORE2[I];
1   15:4   213           CORE2[I]:=CORE2[I-1];
1   15:4   247           CORE2[I-1]:=TEMP2;
1   15:4   268           IF I>2 THEN
1   15:5   275             I:=I-1;
1   15:3   283           END
1   15:2   283         ELSE
1   15:3   285           I:=I+1;
1   15:1   293       UNTIL I>NCORELAST;
1   15:0   302     END;
1   15:0   320
```

SORTCOREFILE constructs the permutation vector for the performance items.

```
334  1   16:D     1 (*$P*)PROCEDURE OPENISSUEINDEX;
335  1   16:0     0   BEGIN
336  1   16:0     0    (*$I-*)
337  1   16:1     0    RESET(ISSUE,ISSUENAME);
338  1   16:1    11    (*$I+*)
339  1   16:1    11    IF IORESULT<>0 THEN
340  1   16:2    17      BEGIN
341  1   16:3    17        WRITELN('Please bear with me while I create the Issue Index on the disk');
342  1   16:3    99        REWRITE(ISSUE,ISSUENAME);
343  1   16:3   112        FOR I:=1 TO 225 DO
344  1   16:4   128          ISSUE^.DATA[I]:=0;
345  1   16:3   167        FOR I:=1 TO NISSUES DO
346  1   16:4   183          BEGIN
347  1   16:5   183            FOR J:=1 TO 2 DO
348  1   16:6   197              ISSUE^.NAME[J]:='                                        ';
349  1   16:5   268            ISSUE^.NUM:=I;
350  1   16:5   275            SEEK(ISSUE,I);
351  1   16:5   286            PUT(ISSUE);
352  1   16:5   294            IF(EOF(ISSUE))THEN
353  1   16:6   304              BEGIN
354  1   16:7   304                WRITELN('OUT OF DISK SPACE');
355  1   16:7   341                WRITELN(' **FATAL ERROR** ');
356  1   16:7   378                ANYKEY;
357  1   16:7   380                SETCHAIN('GREETING');
358  1   16:7   394                EXIT(PROGRAM);
359  1   16:6   398                END;
360  1   16:4   398            END;
361  1   16:3   408        CLOSE(ISSUE,LOCK);
362  1   16:3   417        OPENISSUEINDEX;
363  1   16:3   419        EXIT(OPENISSUEINDEX);
364  1   16:2   423        END
365  1   16:1   423      ELSE
366  1   16:2   425        BEGIN
367  1   16:3   425          NUISSUES:=NISSUES+1;
368  1   16:3   433          REPEAT
369  1   16:4   433            NUISSUES:=NUISSUES-1;
370  1   16:4   441            SEEK(ISSUE,NUISSUES);
371  1   16:4   452            GET(ISSUE)
372  1   16:3   460            UNTIL (ISSUE^.NAME[1]<>'                                   ') OR (NUISSUES= 1);
373  1   16:3   527          IF (NUISSUES=1) AND (COPY(ISSUE^.NAME[1],1,5)='     ') THEN
374  1   16:4   567            NUISSUES:=0;
375  1   16:2   571          END;
376  1   16:0   571      END;
377  1   16:0   598
```

OPENISSUEINDEX counts how many measurement purposes were specified in previous analyses. If ISSUE file does not exist, it creates one.

```
 1    17:D      1 (*$P*)PROCEDURE DISPLAYNAME;
 1    17:0      0   BEGIN
 1    17:1      0     SEEK(ISSUE,I);
 1    17:1     11     GET(ISSUE);
 1    17:1     19     WRITELN(I,'.  ',ISSUE^.NAME[1],CHR(13),'      ',ISSUE^.NAME[2],CHR(13));
 1    17:0    135   END;
 1    17:0    148
```

DISPLAYNAME displays the name of one measurement purpose.

```
1    18:D      1 (*$P*)PROCEDURE DISPLAYISSUES;
1    18:0      0   BEGIN
1    18:1      0     PAGE(OUTPUT);
1    18:1     10     IF NUISSUES=0 THEN
1    18:2     17       BEGIN
1    18:3     17         WRITELN('Currently, there are no measurement purposes in the APM for this
                         system and subsystem');
1    18:2    146         END
1    18:1    146       ELSE
1    18:2    148         BEGIN
1    18:3    148           WRITELN('The following measurement purposes are currently included ');
1    18:3    238           FOR I:=1 TO NUISSUES DO
1    18:4    254             BEGIN
1    18:5    254               DISPLAYNAME;
1    18:5    256               IF (I MOD 6=0) THEN
1    18:6    265                 BEGIN
1    18:7    265                   ANYKEY;
1    18:7    267                   PAGE(OUTPUT);
1    18:6    277                   END;
1    18:4    277               END;
1    18:2    287         END;
1    18:0    287     END;
1    18:0    306
```

DISPLAYISSUES displays names of all measurement purposes.

```
1   19:D     1 (*$P*)PROCEDURE ERASEAFASTISSUE(III:INTEGER);
1   19:0     0   BEGIN;
1   19:1     0     RESET(FASTISSUE,NAMEFASTISSUE);
1   19:1    13     IF IORESULT= 0 THEN
1   19:2    19       BEGIN
1   19:3    19         SEEK(FASTISSUE,III);
1   19:3    28         FOR J:=1 TO 300 DO
1   19:4    44           FASTISSUE^.PRINTIT[J]:=FALSE;
1   19:3    72         PUT(FASTISSUE);
1   19:3    80         CLOSE(FASTISSUE);
1   19:2    89         END;
1   19:0    89     END;
1   19:0   104
```

ERASEAFASTISSUE: FASTISSUE must be erased for any measurement purpose being modified. ERASEAFASTISSUE does this erasure.

```
1   20:D     1 (*$P*)PROCEDURE ADDISSUE(AI:INTEGER);
1   20:0     0   BEGIN
1   20:1     0     SEEK(ISSUE,AI);
1   20:1     9     GET(ISSUE);
1   20:1    17     WRITELN('Please describe the new measurement purpose in 2 68-character lines');
1   20:1   104     FOR I:=1 TO 2 DO
1   20:2   118       BEGIN
1   20:3   118         WRITELN('Please type line #',I,':');
1   20:3   178         REPEAT
1   20:4   178           INLINE;
1   20:4   180           IF LENGTH(LINER)>68 THEN
1   20:5   189             WRITELN('Line contains over 68 characters, please retype');
1   20:3   256           UNTIL LENGTH(LINER)<=68;
1   20:3   265         ISSUE^.NAME[I]:=LINER;
1   20:2   285         END;
1   20:1   295     SEEK(ISSUE,AI);
1   20:1   304     PUT(ISSUE);
1   20:0   312     END;
1   20:0   330
```

ADDISSUE adds a measurement purpose.

```
1   21:D     1 (*$P*)PROCEDURE REMOVEISSUE(RI:INTEGER);
1   21:0     0   BEGIN
1   21:1     0     SEEK(ISSUE,RI);
1   21:1     9     GET(ISSUE);
1   21:1    17     FOR J:=1 TO 2 DO
1   21:2    31       ISSUE^.NAME[J]:='                                   ';
1   21:1   102     FOR J:=1 TO 225 DO
1   21:2   118       ISSUE^.DATA[J]:=0;
1   21:1   157     SEEK (ISSUE,RI);
1   21:1   166     PUT(ISSUE);
1   21:1   174     IF RI=NUISSUES THEN
1   21:2   181       NUISSUES:=NUISSUES-1;
1   21:1   189     ERASEAFASTISSUE(RI);
1   21:0   192     END;
1   21:0   208
```

REMOVEISSUE removes a measurement purpose from the measurement purpose list.

```
 1   22:D     1 (*$P*)PROCEDURE ALTERISSUES;
 1   22:0     0   BEGIN
 1   22:1     0     DISPLAYISSUES;
 1   22:1     2     GOTOXY(0,16);
 1   22:1     7     WRITE(CHR(11));
 1   22:1    17     WRITE('You may perform any of the following procedures:',chr(13),
 1   22:1    87 '  1. Analyze a measurement purpose  2. Specify a new measurement purpose',chr(13),
 1   22:1   181 '  3. Remove a measurement purpose   4. Replace a measurement purpose',chr(13),
 1   22:1   271 '  5. Pack meas purposes efficiently 6. Select a different analytic proc.',chr(13),
 1   22:1   365 'Please select one: ');
 1   22:1   396     REPEAT
 1   22:2   396       KEYN;
 1   22:2   398       IF (I<1) OR (I>6) THEN
 1   22:3   411         WRITELN('Please type an integer between 1 and 5');
 1   22:1   469       UNTIL (I>=1) AND (I<=6);
 1   22:1   482     CASE I OF
 1   22:1   487       1: EXAMINEISSUES;
 1   22:1   491       2: BEGIN
 1   22:3   491           GOTOXY(0,16);
 1   22:3   496           WRITE(CHR(11));
 1   22:3   506           IF NUISSUES>=NISSUES THEN
 1   22:4   515             BEGIN
 1   22:5   515               WRITELN('ISSUE INDEX IS FULL--NO ADDITIONAL ISSUES CAN BE ADDED');
 1   22:5   589               ANYKEY;
 1   22:4   591               END
 1   22:3   591             ELSE
 1   22:4   593               BEGIN
 1   22:5   593                 NUISSUES:=NUISSUES+1;
 1   22:5   601                 ADDISSUE(NUISSUES);
 1   22:4   606                 END;
 1   22:2   606           END;
 1   22:1   608         3: BEGIN
 1   22:3   608             WRITE('Which one (type 0 when done)? ');
 1   22:3   650             KEYN;
 1   22:3   652             IF I<>0 THEN
 1   22:4   659               REMOVEISSUE(I);
 1   22:2   664             END;
 1   22:1   666         4: BEGIN
 1   22:3   666             WRITE('Which one (type 0 when done)?');
 1   22:3   707             KEYN;
```

ALTERISSUES presents menu of options showing what analyst can do with measurement purposes.

```
 1    22:3    709            IF I<>0 THEN
 1    22:4    716              BEGIN
 1    22:5    716                GOTOXY(0,16);
 1    22:5    721                WRITE(CHR(11));
 1    22:5    731                PREPKEY(73,'Is this merely an improvement in the descriptor? ');
 1    22:5    798                IF ANS='Y' THEN
 1    22:6    805                  BEGIN
 1    22:7    805                    ADDISSUE(I);
 1    22:6    810                    END
 1    22:5    810                  ELSE
 1    22:6    812                    BEGIN
 1    22:7    812                      REMOVEISSUE(I);
 1    22:7    817                      NUISSUES:=NUISSUES+1;
 1    22:7    825                      ADDISSUE(I);
 1    22:6    830                      END;
 1    22:4    830                END;
 1    22:2    830             END;
 1    22:1    832       5: COMPACTISSUES;
 1    22:1    836       6: BEGIN
 1    22:3    836             BRANCHOUT;
 1    22:3    838             SETCHAIN('GREETING');
 1    22:3    852             EXIT(PROGRAM);
 1    22:2    856             END;
 1    22:1    858       END;
 1    22:1    878     ALTERISSUES;
 1    22:0    880   END;
 1    22:0    898
```

See previous page for program description.

```
1   23:D     1 (*$P*)PROCEDURE GETINDEX;
1   23:0     0   BEGIN
1   23:1     0     PAGE(OUTPUT);
1   23:1    10     WRITELN('Please be patient...',chr(13),' I am preparing the computer for you');
1   23:1   109       BEGIN
1   23:2   109         SEEK(ISSUE,NCURISSUE);
1   23:2   120         GET(ISSUE);
1   23:2   128         FOR I:=1 TO 225 DO
1   23:3   144         BEGIN
1   23:4   144           ISSUEDATA[I]:=ISSUE^.DATA[I];
1   23:3   186           END;
1   23:1   196         END;
1   23:0   196     END;
1   23:0   210
```

GETINDEX places reference to performance item into array ISSUEDATA or the measurement purpose currently being processed.

```
1   24:D     1 (*$P*)PROCEDURE OPENDATAFILE;
1   24:0     0   BEGIN
1   24:0     0     (*$I-*)
1   24:1     0     RESET(DATANODE,DATANAME);
1   24:1    10     (*$I+*)
1   24:1    10     I:=IORESULT;
1   24:1    15     IF I<>0 THEN
1   24:2    22       BEGIN
1   24:3    22         WRITELN('DATABASE MUST BE CREATED BEFORE ISSUES ARE LINKED TO DATABASE');
1   24:3   103         ANYKEY;
1   24:3   105         BRANCHOUT;
1   24:3   107         SETCHAIN('GREETING');
1   24:3   121         EXIT(PROGRAM);
1   24:2   125         END;
1   24:0   125     END;
1   24:0   138
```

OPENDATAFILE verifies the presence of performance items.

```
1   25:D     1 (*$P*)PROCEDURE TOPSCREEN;
1   25:0     0     BEGIN
1   25:1     0       GOTOXY(0,4);
1   25:1     5       WRITE(CHR(11));
1   25:1    15       NSCREEN:=3;
1   25:1    19       M:=LENGTH(CURSYS);
1   25:1    27       IF M>16 THEN
1   25:2    34         M:=16;
1   25:1    38       LINE:=COPY(CURSYS,1,M);
1   25:1    57       WRITE('$',LINE,' Systems');
1   25:1    99       GOTOXY(26,4);
1   25:1   104       M:=LENGTH(CURSP);
1   25:1   112       IF M>16 THEN
1   25:2   119         M:=16;
1   25:1   123       LINE:=COPY(CURSP,1,M);
1   25:1   142       WRITE('$',LINE);
1   25:1   164       GOTOXY(44,4);
1   25:1   169       M:=LENGTH(CURSUB);
1   25:1   177       IF M>16 THEN
1   25:2   184         M:=16;
1   25:1   188       LINE:=COPY(CURSUB,1,M);
1   25:1   207       WRITELN('$',LINE);
1   25:1   237       GOTOXY(62,4);
1   25:1   242       WRITELN('$',PAC);
1   25:1   272       M:=LENGTH(XOBJECTIVE);
1   25:1   280       IF M>67 THEN M:=67;
1   25:1   291       LINE:=COPY(XOBJECTIVE,1,M);
1   25:1   310       IF NSCREEN>1 THEN
1   25:2   317         WRITELN('Objective[',NOBJECTIVE,']:',LINE);
1   25:1   385       M:=LENGTH(XFUNPUR);
1   25:1   393       IF M>67 THEN M:=67;
1   25:1   404       LINE:=COPY(XFUNPUR,1,M);
1   25:1   423       IF NSCREEN>2 THEN
1   25:2   430         WRITELN('Fctl Prps[',NFUNPUR,']:',LINE);
1   25:1   498       WRITELN(' ');
1   25:0   516       END;
1   25:0   528
```

TOPSCREEN produces the header material on the display screen.

```
 1   26:D     1   (*$P*)PROCEDURE SAVEINDEX;
 1   26:0     0   BEGIN
 1   26:1     0     PAGE(OUTPUT);
 1   26:1    10     WRITELN('Please be patient...',chr(13),'   I am saving all of your hard work');
 1   26:1   108     SEEK(ISSUE,NCURISSUE);
 1   26:1   119     GET(ISSUE);
 1   26:1   127     FOR I:=1 TO 225 DO
 1   26:2   143       BEGIN
 1   26:3   143         ISSUE^.DATA[I]:=ISSUEDATA[I];
 1   26:2   185         END;
 1   26:1   195     SEEK(ISSUE,NCURISSUE);
 1   26:1   206     PUT(ISSUE);
 1   26:0   214     END;
 1   26:0   228
```

SAVEINDEX records the references to performance items for a given measurement purpose in the issuedata file for use by other programs.

```
 1    3:D     1 (*$P*)PROCEDURE COMPACTISSUES;
 1    3:0     0   BEGIN
 1    3:1     0     FOR J:=1 TO NISSUES DO
 1    3:2    16       ERASEAFASTISSUE(J);
 1    3:1    31     M:=0;
 1    3:1    35     I:=0;
 1    3:1    39     REPEAT
 1    3:2    39       I:=I+1;
 1    3:2    47       REPEAT
 1    3:3    47         J:=I+M;
 1    3:3    57         SEEK(ISSUE,J);
 1    3:3    68         GET(ISSUE);
 1    3:3    76         IF (COPY(ISSUE^.NAME[1],1,5)='     ') THEN
 1    3:4   110           M:=M+1;
 1    3:2   118         UNTIL (COPY(ISSUE^.NAME[1],1,5)<>'     ') OR (J>NUISSUES);
 1    3:2   160       IF J<=NUISSUES THEN
 1    3:3   169         BEGIN
 1    3:4   169           J:=I+M;
 1    3:4   179           SEEK(ISSUE,I);
 1    3:4   190           PUT(ISSUE);
 1    3:3   198           END;
 1    3:1   198       UNTIL J>=NUISSUES;
 1    3:1   207     I:=I+1;
 1    3:1   215     FOR K:=I TO NISSUES DO
 1    3:2   233       REMOVEISSUE(K);
 1    3:1   248     IF M>0 THEN NUISSUES:=NUISSUES-M+1;
 1    3:0   267     END;
 1    3:0   288
 1    3:0   288   (*$I #5:MEASPURP2.TEXT*)
 1    3:0   288
```

COMPACTISSUES packs measurement purpose references more efficiently.

```
1   27:D     1 (*$P*)PROCEDURE REFISSUE;
1   27:0     0   BEGIN
1   27:1     0     J:=0;
1   27:1     4     REPEAT
1   27:2     4       J:=J+1;
1   27:1    12       UNTIL(ISSUEDATA[J]=0) OR (J=224);
1   27:1    50     IF (J=224) AND (ISSUEDATA[J]<>0) THEN
1   27:2    88       BEGIN
1   27:3    88         WRITELN('SORRY--BUT YOU ALREADY HAVE 224 REFERENCES FOR THIS MEASUREMENT
                         PURPOSE',CHR(13),
1   27:3   181                 '  SO YOU CAN NOT ADD ANOTHER ONE!!!');
1   27:3   236         ANYKEY;
1   27:3   238         EXIT(REFISSUE)
1   27:2   242         END
1   27:1   242       ELSE
1   27:2   244         BEGIN
1   27:3   244           ISSUEDATA[J]:=CORE[I];
1   27:3   284           FLAG[I]:=TRUE;
1   27:2   301           END;
1   27:0   301     END;
1   27:0   318
```

REFISSUE adds a new performance item reference to the measurement purpose index.

```
1   28:D      1 (*$P*)PROCEDURE UNREFISSUE;
1   28:0      0   BEGIN
1   28:1      0     K:=0;
1   28:1      4     REPEAT
1   28:2      4       K:=K+1;
1   28:1     12       UNTIL (K=224) OR (ISSUEDATA[K]=CORE[I]);
1   28:1     63     IF K>=224 THEN
1   28:2     72       BEGIN
1   28:3     72         WRITELN('ERROR--FLAG SAYS REFERENCED, ISSUEDATA SAYS UNREFERENCED');
1   28:3    148         EXIT(UNREFISSUE);
1   28:2    152         END;
1   28:1    152     J:=K-1;
1   28:1    160     REPEAT
1   28:2    160       J:=J+1;
1   28:2    168       ISSUEDATA[J]:=ISSUEDATA[J+1];
1   28:1    210       UNTIL (ISSUEDATA[J]=0) OR (J=224);
1   28:1    248     ISSUEDATA[225]:=0;
1   28:1    275     FLAG[I]:=FALSE;
1   28:0    292     END;
1   28:0    308
```

UNREFISSUE removes a reference to a performance item from a measurement purpose index.

```
1   29:D     1 (*$P*)PROCEDURE REWORD;
1   29:0     0   BEGIN
1   29:1     0     ADDISSUE(NCURISSUE);
1   29:0     5     END;
1   29:0    18
```

REWORD allows one to reword a measurement purpose label.

```
1   30:D     1 (*$P*)PROCEDURE SETUPFLAG;
1   30:0     0   BEGIN
1   30:1     0     FOR K:=1 TO 300 DO
1   30:2    16       FLAG[K]:=FALSE;
1   30:1    43     IF ISSUEDATA[1]<>0 THEN
1   30:2    71       BEGIN
1   30:3    71         WRITELN('Please be patient...', chr(13),
1   30:3   113                 '  I am setting up your measurement purpose');
1   30:3   175         FOR K:=1 TO 225 DO
1   30:4   191           BEGIN
1   30:5   191             IF ISSUEDATA[K]<>0 THEN
1   30:6   221               BEGIN
1   30:7   221                 NODE:=0;
1   30:7   225                 REPEAT
1   30:8   225                   NODE:=NODE+1;
1   30:8   233                   IF ISSUEDATA[K]=CORE[NODE] THEN
1   30:9   276                     FLAG[NODE]:=TRUE;
1   30:7   293                   UNTIL (NODE=300);
1   30:6   302                 END;
1   30:4   302               END;
1   30:2   312         END;
1   30:0   312     END;
1   30:0   332
```

SETUPFLAG sets up a flag for each performance item which belongs to a measurement purpose.

```
1   31:D     1 (*$P*)PROCEDURE GOEXAMINE;
1   31:0     0   BEGIN
1   31:1     0     OK:=FALSE;
1   31:1     4     REPEAT
1   31:2     4       GOTOXY(0,16);
1   31:2     9       WRITE(CHR(11),'Which one would you like to analyze(type 0 to reconsider)?');
1   31:2    89       KEYN;
1   31:2    91       NCURISSUE:=I;
1   31:2    97       IF (I>NUISSUES) OR (I<0) THEN
1   31:3   112         WRITELN('Please type an integer between 1 and ',NUISSUES,':');
1   31:1   191       UNTIL (I<=NUISSUES) AND (I>=0);
1   31:1   206     IF I=0 THEN
1   31:2   213       EXIT(EXAMINEISSUES);
1   31:1   217     ERASEAFASTISSUE(I);
1   31:1   222     GETINDEX;
1   31:1   224     SETUPFLAG;
1   31:1   226     PAGE(OUTPUT);
1   31:1   236     WRITELN('You have chosen to analyze measurement purpose: ',NCURISSUE);
1   31:1   316     I:=NCURISSUE;
1   31:1   322     DISPLAYNAME;
1   31:1   324     GOTOXY(0,3);
1   31:1   329     WRITELN(CHR(26),'3Black on white',chr(26),
1   31:1   376             '2 performance items are associated with the measurement purpose');
1   31:1   459     RESET(DATANODE,DATANAME);
1   31:0   471     END;
1   31:0   486
```

GOEXAMINE determines which measurement purpose the analyst wishes to analyze.

```
1   32:D    1 (*$P*)PROCEDURE REVERSEISSUES;
1   32:0    0   BEGIN
1   32:1    0     IF FLAG[I]=TRUE
1   32:1   16       THEN UNREFISSUE
1   32:1   21       ELSE REFISSUE;
1   32:1   27       COUNT:=0;
1   32:0   31     END;
1   32:0   44
```

REVERSEISSUES—if analyst wishes to add a performance item to the measurement purpose, reverse issues calls REFISSUE. If analyst wishes to remove a performance item from the measurement purpose, REVERSEISSUES calls UNREFISSUE.

```
1   33:D     1 (*$P*)PROCEDURE ENDPAGE;
1   33:0     0   BEGIN
1   33:1     0     I:=0;
1   33:1     4     GOTOXY(0,19);
1   33:1     9     WRITE(CHR(11));
1   33:1    19     PREPKEY(94,'Change assns between measurement purpose and a performance item?');
1   33:1    95     IF ORD(ANS)=27 THEN
1   33:2   102       BEGIN
1   33:3   102         CLOSE(DATANODE);
1   33:3   110         SAVEINDEX;
1   33:3   112         EXIT(EXAMINEISSUES);
1   33:2   116         END;
1   33:1   116     IF ANS='Y' THEN
1   33:2   123       BEGIN
1   33:3   123         GOTOXY(0,19);
1   33:3   128         WRITE(CHR(11));
1   33:3   138         WRITE('Which one (type 0 if none; 999 if all)? ');
1   33:3   190         KEYN;
1   33:3   192         I:=I+GOPAGE-1;
1   33:3   204         IF (I>0) AND (I<300)THEN
1   33:4   219           IF (CHARACTERISTIC=TRUE) THEN
1   33:5   227             REVERSEISSUES
1   33:4   227             ELSE
1   33:5   231               BEGIN
1   33:6   231                 GOTOXY(0,21);
1   33:6   236                 WRITE(CHR(11));
1   33:6   246                 WRITELN('ERROR--PERFORMANCE ITEM ',I,'IS NOT A CHARACTERISTIC!');
1   33:5   338                 END;
1   33:3   338         N:=999+GOPAGE-1;
1   33:3   350         IF I=N THEN
1   33:4   359           FOR I:=GOPAGE TO NODE DO
1   33:5   377             IF(CORE[I]-CORE[I] DIV 100 * 100<>0) THEN
1   33:6   447               REVERSEISSUES;
1   33:3   459         NODE:=GOPAGE-1;
1   33:2   467         END
1   33:1   467       ELSE
1   33:2   469         GOPAGE:=NODE+1;
1   33:1   477     GOTOXY(0,8);
1   33:1   482     WRITE(CHR(11));
1   33:0   492     END;
1   33:0   508
```

ENDPAGE displays the "do you want to change association" message and then it processes the response.

```
1   34:D     1 (*$P*)PROCEDURE CHANGETOPSCREEN;
1   34:0     0   BEGIN
1   34:1     0     IF CORE[NODE]=0 THEN
1   34:2    30       EXIT(CHANGETOPSCREEN);
1   34:1    34     SEEK(DATANODE,CORE2[NODE]);
1   34:1    57     GET(DATANODE);
1   34:1    64     IF CORE[NODE] DIV 10000 * 10000 = CORE[NODE] THEN
1   34:2   129       BEGIN
1   34:3   129         I:=TRUNC(CORE[NODE] DIV 1000000);
1   34:3   183         PAC:=ASPECT[I];
1   34:3   201         NPAC:=I;
1   34:3   207         XOBJECTIVE:=DATANODE^.TAXA;
1   34:3   215         NOBJECTIVE:=DATANODE^.NTAXA[2];
1   34:2   228         END
1   34:1   228       ELSE
1   34:2   230         BEGIN
1   34:3   230           XFUNPUR:=DATANODE^.TAXA;
1   34:3   238           NFUNPUR:=DATANODE^.NTAXA[3];
1   34:2   251           END;
1   34:1   251     IF(NEXTCHARACTERISTIC=TRUE) THEN
1   34:2   259       BEGIN
1   34:3   259         TOPSCREEN;
1   34:3   261         COUNT:=0;
1   34:3   265         GOTOXY(0,8);
1   34:3   270         WRITE(CHR(11));
1   34:2   280         END;
1   34:0   280     END;
1   34:0   292
```

CHANGETOPSCREEN changes contents of the header printed at the top of each page.

```
1   35:D     1 (*$P*)PROCEDURE ONEPERFITEMDISPLAY;
1   35:0     0   BEGIN
1   35:1     0     SEEK(DATANODE,CORE2[NODE]);
1   35:1    23     GET(DATANODE);
1   35:1    30     K:=DATANODE^.NTAXA[M];
1   35:1    45     LLENGTH:=72;
1   35:1    49     LINE:=DATANODE^.TAXA;
1   35:1    57     IF K<>0 THEN
1   35:2    64       BEGIN
1   35:3    64         INVERSE:=2;
1   35:3    68         NN:=0;
1   35:3    72         CASE M OF
1   35:3    77           1:TEMPL5:=1000000;
1   35:3   114           2:TEMPL5:=10000;
1   35:3   133           3:TEMPL5:=100;
1   35:3   150           4:TEMPL5:=1;
1   35:3   167           END;
1   35:3   182         TEMPL6:=CORE[NODE] DIV TEMPL5;
1   35:3   220         REPEAT
1   35:4   220           NN:=NN+1;
1   35:3   228           UNTIL (NN=225) OR (ISSUEDATA[NN] DIV TEMPL5 = TEMPL6);
1   35:3   277         IF NN<225 THEN
1   35:4   286           INVERSE:=3;
1   35:3   290         WRITE('  ',CHR(26),INVERSE,K,'. ');
1   35:3   352         INDENT:=6;
1   35:3   356         SHOWALINE;
1   35:3   358         WRITELN(CHR(26),'2');
1   35:2   386         END;
1   35:0   386     END;
1   35:0   402
```

ONEPERFITEMDISPLAY displays one performance item in the body of the display used to select which performance item ought to be processed next.

```
1   36:D     1 {$$P$}PROCEDURE SHOWPERFITEMS;
1   36:0     0   BEGIN
1   36:1     0     OK:=FALSE;
1   36:1     4     DISPCOUNT:=0;
1   36:1     8     IF M=2 THEN TEMPL1:=1000000;
1   36:1    50     IF M=2 THEN TEMPL3:=10000;
1   36:1    74     IF M=3 THEN TEMPL1:=10000;
1   36:1    98     IF M=3 THEN TEMPL3:=100;
1   36:1   120     TEMPL2:=TEMP DIV TEMPL1;
1   36:1   146     FOR NODE:=1 TO NCORELAST DO
1   36:2   162       BEGIN
1   36:3   162         IF (TEMPL2=CORE[NODE] DIV TEMPL1) AND
1   36:3   201          (CORE[NODE] DIV TEMPL3 * TEMPL3 = CORE[NODE]) THEN
1   36:4   265           BEGIN
1   36:5   265             OK:=TRUE;
1   36:5   269             ONEPERFITEMDISPLAY;
1   36:5   271             DISPCOUNT:=DISPCOUNT+1;
1   36:4   279             END;
1   36:3   279         IF (DISPCOUNT DIV 15 * 15=DISPCOUNT) AND (DISPCOUNT<>0) THEN
1   36:4   298           BEGIN
1   36:5   298             DISPCOUNT:=0;
1   36:5   302             ANYKEY;
1   36:5   304             GOTOXY(0,2);
1   36:5   309             WRITE(CHR(11));
1   36:4   319             END;
1   36:2   319         END;
1   36:1   329       IF OK=FALSE THEN
1   36:2   337         WRITELN('   ...none');
1   36:0   367       END;
1   36:0   384
```

SHOWPERFITEMS controls production of the body of displays of performance items.

```
1   37:D     1 (*$P*)PROCEDURE SPECIFYSTART;
1   37:0     0   BEGIN
1   37:1     0     REPEAT
1   37:2     0       GOTOXY(0,4);
1   37:2     5       WRITE(CHR(11));
1   37:2    15       WRITELN('The following aspects are part of the APM:');
1   37:2    77       FOR N:=1 TO 5 DO
1   37:3    91         INVERSEA[N]:=2;
1   37:2   116       FOR NODE:=1 TO 300 DO
1   37:3   132         IF FLAG[NODE]=TRUE THEN
1   37:4   153           BEGIN
1   37:5   153             N:=TRUNC(CORE[NODE] DIV 1000000);
1   37:5   207             IF N>0 THEN
1   37:6   214               INVERSEA[N]:=3;
1   37:4   229             END;
1   37:2   239       FOR N:=1 TO 5 DO
1   37:3   253         WRITELN(' ',CHR(26),INVERSEA[N],N,'. ',ASPECT[N],CHR(26),'2');
1   37:2   382       WRITE('Please select one: ');
1   37:2   413       KEYN;
1   37:2   415       PAC:=ASPECT[I];
1   37:2   433       NPAC:=I;
1   37:2   439       TEMPL4:=I;
1   37:2   456       TEMP:=TEMPL4*1000000;
1   37:2   501       GOTOXY(0,4);
1   37:2   506       WRITE(CHR(11));
1   37:2   516       M:=2;
1   37:2   520       WRITELN('The following Objectives are available for the aspect selected: ');
1   37:2   604       SHOWPERFITEMS;
1   37:2   606       WRITE('Please select one: ');
1   37:2   637       KEYN;
1   37:2   639       NOBJECTIVE:=I;
1   37:2   645       TEMPL4:=I;
1   37:2   662       TEMP:=TEMP+TEMPL4*10000;
1   37:2   699       FOR I:=1 TO 300 DO
1   37:3   715         IF TEMP=CORE[I] THEN
1   37:4   746           BEGIN
1   37:5   746             SEEK(DATANODE,CORE2[NODE]);
1   37:5   769             GET(DATANODE);
1   37:5   776             XOBJECTIVE:=DATANODE^.TAXA;
1   37:4   784             END;
```

SPECIFYSTART allows analyst to select where he/she wants to start analyzing measurement purpose links to performance items.

```
902   1   37:2   794      GOTOXY(0,4);
903   1   37:2   799      WRITE(CHR(11));
904   1   37:2   809      N:=3;
905   1   37:2   813      WRITELN('The following Functional Purposes are available for the objective
                          selected: ');
906   1   37:2   909      SHOWPERFITEMS;
907   1   37:2   911      WRITE('Please select one: ');
908   1   37:2   942      KEYN;
909   1   37:2   944      TEMPL4:=I;
910   1   37:2   961      TEMP:=TEMP+TEMPL4*100;
911   1   37:2   996      FOR I:=1 TO 300 DO
912   1   37:3  1012        IF CORE[I]=TEMP THEN
913   1   37:4  1043          BEGIN
914   1   37:5  1043            NODE:=I-1;
915   1   37:5  1051            GOPAGE:=NODE+1;
916   1   37:4  1059            END;
917   1   37:2  1069      IF NODE=0 THEN
918   1   37:3  1076        WRITELN('Performance item # ',temp,' does not exist!');
919   1   37:1  1163      UNTIL NODE<>0;
920   1   37:0  1170      END;
921   1   37:0  1196
```

See previous page for program description.

```
1    2:D     1 (*$P*)PROCEDURE EXAMINEISSUES;
1    2:0     0   BEGIN
1    2:1     0     GOEXAMINE;
1    2:1     2     IF OK=TRUE THEN
1    2:2    10       EXIT(EXAMINEISSUES);
1    2:1    14     LASTCHARACTERISTIC:=TRUE;
1    2:1    18     IF CORE[NCORELAST]-CORE[NCORELAST] DIV 100 * 100 = 0 THEN
1    2:2    88       LASTCHARACTERISTIC:=FALSE;
1    2:1    92     NODE:=0;
1    2:1    96     GOPAGE:=1;
1    2:1   100     PREPKEY(222,'Do you wish to analyze the first performance item?');
1    2:1   158     IF ANS='N' THEN
1    2:2   165       SPECIFYSTART;
1    2:1   167     REPEAT
1    2:2   167       REPEAT
1    2:3   167         NODE:=NODE+1;
1    2:3   175         CHARACTERISTIC:=TRUE;
1    2:3   179         IF CORE[NODE]-CORE[NODE] DIV 100 * 100 = 0 THEN
1    2:4   249           CHARACTERISTIC:=FALSE;
1    2:3   253         NEXTCHARACTERISTIC:=TRUE;
1    2:3   257         IF CORE[NODE+1]-CORE[NODE+1] DIV 100 * 100 = 0 THEN
1    2:4   331           NEXTCHARACTERISTIC:=FALSE;
1    2:3   335         IF(CHARACTERISTIC=FALSE)THEN
1    2:4   343           CHANGETOPSCREEN;
1    2:3   345         IF CHARACTERISTIC=FALSE THEN
1    2:4   353           GOPAGE:=GOPAGE+1;
1    2:2   361         UNTIL (CHARACTERISTIC=TRUE) OR (NODE>=NCORELAST);
1    2:2   377       IF (NCORELAST<=GOPAGE) AND (LASTCHARACTERISTIC=FALSE)
1    2:2   390         THEN BEGIN
1    2:4   393           CLOSE (DATANODE);
1    2:4   401           SAVEINDEX;
1    2:4   403           EXIT(EXAMINEISSUES);
1    2:3   407           END;
1    2:2   407       INVERSE:=2;
1    2:2   411       SEEK(DATANODE,CORE2[NODE]);
1    2:2   434       GET(DATANODE);
1    2:2   441       IF FLAG[NODE]=TRUE THEN
1    2:3   462         INVERSE:=3;
1    2:2   466       L:=NODE-GOPAGE+1;
1    2:2   478       WRITE(CHR(26),INVERSE,'  ',L,'.  [');
```

EXAMINEISSUES does initial setup for analyzing measurement purpose-performance item links.

```
1   2:2  542       FOR K:=1 TO 4 DO
1   2:3  556         BEGIN
1   2:4  556           J:=DATANODE^.NTAXA[K];
1   2:4  571           WRITE(J,'.');
1   2:3  593           END;
1   2:2  603       WRITE(']');
1   2:2  613       LINE:=DATANODE^.TAXA;
1   2:2  621       WRITE(' ');
1   2:2  631       LLENGTH:=60;
1   2:2  635       INDENT:=14;
1   2:2  639       SHOWALINE;
1   2:2  641       WRITELN(CHR(26),'2');
1   2:2  669       COUNT:=COUNT+1;
1   2:2  677       IF (COUNT=5) OR (NODE=NCORELAST) OR
1   2:2  690         (CORE[NODE] DIV 100<>CORE[NODE+1] DIV 100)THEN
1   2:3  754           ENDPAGE;
1   2:1  756       UNTIL (I=0) AND (NODE=NCORELAST);
1   2:1  771     CLOSE(DATANODE);
1   2:1  779     SAVEINDEX;
1   2:0  781     END;
1   2:0  800
1   2:0  800
1   2:0  800
1   2:0  800
1   2:0  800   (*$I #5:MEASPURP2.TEXT*)
1   2:0  800
```

See previous page for program description.

```
1:0      (*$P*)BEGIN
1:0    0 (*$N+*)
1:1    0 INLINECALL:=0;
1:1   62 NISSUES:=5;
1:1   66 BRANCHIN;
1:1   68 DEFINEASPECTS;
1:1   70 APMDSK:=CONCAT(COPY(CURSYS,1,2),COPY(CURSP,1,2),COPY(CURSUB,1,2),':');
1:1  162 CORENAME:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'CO');
1:1  265 NAMEFASTISSUE:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'FA');
1:1  368 WRITELN('I am now sorting your performance items.');
1:1  428 READCOREFILE;
1:1  430 SORTCOREFILE;
1:1  432 ISSUENAME:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'IS');
1:1  535 DATANAME:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'FI');
1:1  638 OPENISSUEINDEX;
1:1  640 ALTERISSUES;
1:0  642 END.
```

Main Program: reads and sorts core file, the index to the performance items. Then, control is transformed to the ALTERISSUES program which presents the list of measurement pruposes and the various analytic procedures which may be performed with measurement purposes.

PRINT

PRINT allows the analyst to print either 1) all performance items, attributes and measures, or 2) performance items, attributes and measures for a given measurement purpose. The analyst may choose to print only some objectives, functional purposes and charactierstics in this program without altering the basic data set.

```
 1     1:D     1 (*$L PRINTER: *)
 1     1:D     1 (*$S+*)
 1     1:D     1 (* Program to print performance items, attribute, and measures list for a given
                    measurement purpo
 1     1:D     1 (* Ronald G. Shapiro   Version 2.0   10/25/82*)
 1     1:D     1 Program Printdatasetts;
 1     1:D     3
28     1:D     3
28     2:D     1    PROCEDURE SETCHAIN(TYTLE:STRING);
28     3:D     1    PROCEDURE SETCVAL(VAL:STRING);
28     4:D     1    PROCEDURE GETCVAL(VAR VAL:STRING);
28     5:D     1    PROCEDURE SWAPON;
28     6:D     1    PROCEDURE SWAPOFF;
28     6:D     1
22     1:D     1
22     1:D     3
22     2:D     3      FUNCTION PADDLE(SELECT: INTEGER): INTEGER;
22     3:D     3      FUNCTION BUTTON(SELECT: INTEGER): BOOLEAN;
22     4:D     1      PROCEDURE TTLOUT(SELECT: INTEGER; DATA: BOOLEAN);
22     5:D     3      FUNCTION KEYPRESS: BOOLEAN;
22     6:D     3      FUNCTION RANDOM: INTEGER;
22     7:D     1      PROCEDURE RANDOMIZE;
22     8:D     1      PROCEDURE NOTE(PITCH,DURATION: INTEGER);
22     8:D     3
 1     1:D     3 Uses Chainstuff,APPLESTUFF;
 1     1:D     3
```

These procedures are part of the Apple Computer's CHAINSTUFF library entry. The demonstration package uses only SETCHAIN which causes another program to be activated.

```
26   1    1:D     3 (*$P*)CONST
27   1    1:D     3   OBJLBL1='The system must be capable of:';
28   1    1:D     3   OBJLBL2='The system must carry out the following activities:';
29   1    1:D     3   OBJLBL3='The system must produce:';
30   1    1:D     3   OBJLBL4='Performance objectives must be met despite:';
31   1    1:D     3   OBJLBL5='Performance objectives must be met despite:';
32   1    1:D     3
```

CONSTANTS are defined.

```
1    1:D     3 (*$P*)TYPE
1    1:D     3   ISSUEFILE =RECORD
1    1:D     3     NUM:INTEGER;
1    1:D     3     NAME:ARRAY[1..2]OF STRING[80];
1    1:D     3     DATA:ARRAY[1..225]OF INTEGER[8];
1    1:D     3     END;
1    1:D     3
1    1:D     3   FASTFILE =RECORD
1    1:D     3     PRINTIT:PACKED ARRAY[1..300]OF BOOLEAN;
1    1:D     3     END;
1    1:D     3
1    1:D     3   PASSFILE =RECORD
1    1:D     3     CURSYS,CURSP,CURSUB,PAC:STRING[80];
1    1:D     3     NCURSYS,NCURSP,NCURSUB,NPAC,FLAG1,FLAG2,FLAG3:INTEGER;
1    1:D     3     END;
1    1:D     3
1    1:D     3   DATABASE =RECORD
1    1:D     3     NTAXA: ARRAY[1..4] OF INTEGER;
1    1:D     3     TAXA: STRING[80];
1    1:D     3     END;
1    1:D     3
1    1:D     3   FILEATTRIBUTES =RECORD
1    1:D     3     NDESCRIPTOR: ARRAY[1..6] OF INTEGER;
1    1:D     3     DESCRIPTOR: STRING[68];
1    1:D     3     END;
1    1:D     3
1    1:D     3   FILEMEASURES =RECORD
1    1:D     3     NDESCRIPTOR: ARRAY[1..6] OF INTEGER;
1    1:D     3     DESCRIPTOR: STRING[68];
1    1:D     3     END;
1    1:D     3
```

ISSUEFILE is a list of measurement purpose names and references to performance items. FASTFILE allows fast processing of measurement purposes. PASSFILE is an inter-program communication. DATABASE is performance item files. FILEATTRIBUTES is attributes file. FILEMEASURES is measures file.

```
1    1:D    3 (*$P*)VAR
1    1:D    3   PASSNODE:FILE OF PASSFILE;
1    1:D  474   DATANODE:FILE OF DATABASE;
1    1:D  819   COREFILE:FILE OF INTEGER[8];
1    1:D 1122   ATTRIBUTES:FILE OF FILEATTRIBUTES;
1    1:D 1463   ATTRFILE:FILE OF INTEGER[12];
1    1:D 1767   MEASURES:FILE OF FILEMEASURES;
1    1:D 2108   MEASFILE:FILE OF INTEGER[12];
1    1:D 2412   ISSUE:FILE OF ISSUEFILE;
1    1:D 3470   FASTISSUE:FILE OF FASTFILE;
1    1:D 3789
1    1:D 3789   CORE:ARRAY[1..300] OF INTEGER[8];
1    1:D 4689   ATTRCORE:ARRAY[1..200] OF INTEGER[12];
1    1:D 5489   MEASCORE:ARRAY[1..400] OF INTEGER[12];
1    1:D 7089   ASPECT:ARRAY[1..5] OF STRING[20];
1    1:D 7144   CORE2:ARRAY[1..300] OF INTEGER;
1    1:D 7444   ATTR2:ARRAY[1..200] OF INTEGER;
1    1:D 7644   MEAS2:ARRAY[1..400] OF INTEGER;
1    1:D 8044   PRINTIT:PACKED ARRAY[1..300] OF BOOLEAN;
1    1:D 8063
1    1:D 8063   XFUNPUR,XOBJECTIVE,PAC,CURSYS,CURSP,CURSUB: STRING[80];
1    1:D 8309   NCURMEASURE,NCURATTRIBUTE,NCURISSUE,
1    1:D 8309     NFUNPUR,NOBJECTIVE,NPAC,NCURSYS,NCURSP,NCURSUB: INTEGER;
1    1:D 8318
1    1:D 8318   ISSUENAME,NAMEATCORE,NAMEATTRIBUTES,NAMEMECORE,NAMEMEASURES: STRING[30];
1    1:D 8398   NAMEFASTISSUE,CORENAME,DATANAME: STRING[30];
1    1:D 8446   LEVEL: STRING[10];
1    1:D 8452   APMDSK:STRING[10];
1    1:D 8458   USERNAME,USERDATE,USERMSG: STRING[80];
1    1:D 8581
1    1:D 8581   TEMP,CORELAST,T1,T2,T3,T4,T5: INTEGER[8];
1    1:D 8602   TEMPX,ATTRLAST,MEASLAST:INTEGER[12];
1    1:D 8614
1    1:D 8614   MODE,INVERSE,HELP,NSCREEN:INTEGER;
1    1:D 8618   NCORELAST,NATTRLAST,NMEASLAST:INTEGER;
1    1:D 8621   NISSUES,NUISSUES,NATTRIBUTES,NMEASURES,NUMEASURES:INTEGER;
1    1:D 8626
1    1:D 8626   I,J,K,L,M,N,CUT,INDENT,COUNT,TEMP2:INTEGER;
1    1:D 8636
1    1:D 8636   NOISSUE,REFERENCED,LONGWAY,DONE,OVER,OK,SKIP,NONE:BOOLEAN;
```

These strings, arrays and variables are used by this program.

```
  1 1:D  8644
  1 1:D  8644   LINER:STRING[4];(*ADDED TO AVOID COMPILER ERROR ON INLINE--NOT USED IN PRINT PGM*)
  1 1:D  8647   ANSWER,LINE:STRING[80];
  1 1:D  8729
  1 1:D  8729   ANS,ANSHOLD: CHAR;
  1 1:D  8731
  1 1:D  8731   PRNT:TEXT;
  1 1:D  9032
  1 2:D     1   PROCEDURE ANYKEY;FORWARD;
  1 3:D     1   PROCEDURE BRANCHIN;FORWARD;
  1 4:D     1   PROCEDURE BRANCHOUT;FORWARD;
  1 5:D     1   PROCEDURE ELIMINATE;FORWARD;
  1 5:D     1
```

Continuation of strings, arrays and variables list from previous page.

```
117   7    1:D     1 (*$P*)SEGMENT PROCEDURE OPENISSUEINDEX;
118   7    1:0     0   BEGIN
119   7    1:1     0     NOISSUE:=FALSE;
120   7    1:1     4     (*$I-*)
121   7    1:1     4     RESET(ISSUE,ISSUENAME);
122   7    1:1    15     (*$I+*)
123   7    1:1    15     IF IORESULT<>0 THEN
124   7    1:2    21       BEGIN
125   7    1:3    21         WRITELN('NO MEAS PURP FILE!');
126   7    1:3    59         NUISSUES:=0;
127   7    1:3    63         NOISSUE:=TRUE;
128   7    1:2    67         END
129   7    1:1    67       ELSE
130   7    1:2    69         BEGIN
131   7    1:3    69           NUISSUES:=NISSUES+1;
132   7    1:3    77           REPEAT
133   7    1:4    77             NUISSUES:=NUISSUES-1;
134   7    1:4    85             SEEK(ISSUE,NUISSUES);
135   7    1:4    96             GET(ISSUE);
136   7    1:3   104             UNTIL(COPY(ISSUE^.NAME[1],1,5)<>'     ') OR (NUISSUES=1);
137   7    1:3   144           IF (NUISSUES=1) AND (COPY(ISSUE^.NAME[1],1,5)='     ') THEN
138   7    1:4   184             NUISSUES:=0;
139   7    1:2   188           END;
140   7    1:1   188       CLOSE(ISSUE);
141   7    1:0   197       END;
142   7    1:0   212
```

OPENISSUEINDEX determines how many measurement purposes there are (if any).

```
  8   1:D     1 (*$P*)SEGMENT PROCEDURE TERMINATE;
  8   1:0     0   BEGIN
  8   1:1     0     WRITELN('PLEASE RUN PROC#2 TO CREATE ATTRIB & MEAS');
  8   1:1    61     ANYKEY;
  8   1:1    64     BRANCHOUT;
  8   1:1    67     SETCHAIN('GREETING');
  8   1:1    81     EXIT(PROGRAM);
  8   1:0    85     END;
  8   1:0    98
```

TERMINATE displays warning message and transfers control back to analytic procedure section of GREETING program.

```
 9     1:D     1 (*$P*)SEGMENT PROCEDURE OPENFASTISSUE;
 9     1:0     0   BEGIN
 9     1:0     0     (*$I-*)
 9     1:1     0     RESET(FASTISSUE,NAMEFASTISSUE);
 9     1:1    11     (*$I+*)
 9     1:1    11     I:=IORESULT;
 9     1:1    16     IF I<>0 THEN
 9     1:2    23       BEGIN
 9     1:3    23         REWRITE(FASTISSUE,NAMEFASTISSUE);
 9     1:3    36         FOR I:=1 TO NISSUES DO
 9     1:4    52           BEGIN
 9     1:5    52             SEEK(FASTISSUE,I);
 9     1:5    63             FOR J:=1 TO 300 DO
 9     1:6    79               FASTISSUE^.PRINTIT[J]:=FALSE;
 9     1:5   107             PUT(FASTISSUE);
 9     1:5   115             IF(EOF(FASTISSUE))THEN
 9     1:6   125               BEGIN
 9     1:7   125                 WRITELN('OUT OF DISK SPACE');
 9     1:7   162                 WRITELN(' **FATAL ERROR** ');
 9     1:7   199                 ANYKEY;
 9     1:7   202                 BRANCHOUT;
 9     1:7   205                 SETCHAIN('GREETING');
 9     1:7   219                 EXIT(PROGRAM);
 9     1:6   223                 END;
 9     1:4   223             END;
 9     1:3   233         CLOSE(FASTISSUE,LOCK);
 9     1:3   242         OPENFASTISSUE;
 9     1:3   244         EXIT(OPENFASTISSUE);
 9     1:2   248         END;
 9     1:1   248     CLOSE(FASTISSUE);
 9     1:0   257     END;
 9     1:0   278
```

OPENFASTISSUE determines whether fastissue file exists. If not, it creates it.

```
10   1:D     1 (*$P*)SEGMENT PROCEDURE READATTRFILE;
10   1:0     0   BEGIN
10   1:0     0   (*$I-*)
10   1:1     0   RESET(ATTRFILE,NAMEATCORE);
10   1:1    11   I:=IORESULT;
10   1:0    16   (*$I+*);
10   1:1    16   IF I<>0 THEN
10   1:2    23     TERMINATE
10   1:1    23     ELSE
10   1:2    28       BEGIN
10   1:3    28         FOR I:=1 TO NATTRIBUTES DO
10   1:4    44           BEGIN
10   1:5    44             GET(ATTRFILE);
10   1:5    52             ATTRCORE[I]:=ATTRFILE^;
10   1:4    80             END;
10   1:3    90         GET(ATTRFILE);
10   1:3    98         ATTRLAST:=ATTRFILE^;
10   1:3   114         NATTRLAST:=TRUNC(ATTRLAST);
10   1:3   127         CLOSE(ATTRFILE);
10   1:2   136         END;
10   1:0   136     END;
10   1:0   150
```

READATTRFILE loads core with index to attributes file.

```
11    1:D     1 (*$P*)SEGMENT PROCEDURE READMEASFILE;
11    1:0     0   BEGIN
11    1:0     0   (*$I-*)
11    1:1     0   RESET(MEASFILE,NAMEMECORE);
11    1:1    11   I:=IORESULT;
11    1:0    16   (*$I+*)
11    1:1    16   IF I<>0 THEN
11    1:2    23     TERMINATE
11    1:1    23     ELSE
11    1:2    28       BEGIN
11    1:3    28         FOR I:=1 TO NMEASURES DO
11    1:4    44           BEGIN
11    1:5    44             GET(MEASFILE);
11    1:5    52             MEASCORE[I]:=MEASFILE^;
11    1:4    80             END;
11    1:3    90         GET(MEASFILE);
11    1:3    98         MEASLAST:=MEASFILE^;
11    1:3   114         NMEASLAST:=TRUNC(MEASLAST);
11    1:3   127         CLOSE(MEASFILE);
11    1:2   136         END;
11    1:0   136     END;
11    1:0   150
```

READMEASFILE loads core with index to measures file.

```
12     1:D      1 (*$P*)SEGMENT PROCEDURE OPENDATAFILE;
12     1:0      0   BEGIN
12     1:0      0     (*$I-*)
12     1:1      0     RESET(DATANODE,DATANAME);
12     1:1     11     (*$I+*)
12     1:1     11     I:=IORESULT;
12     1:1     16     IF I<>0 THEN
12     1:2     23       BEGIN
12     1:3     23         WRITE('DATABASE MUST BE CREATED BEFORE IT CAN BE PRINTED',CHR(13),
12     1:3     94               '... ALSO');
12     1:3    114         TERMINATE;
12     1:2    117         END;
12     1:1    117     CLOSE(DATANODE);
12     1:0    126     END;
12     1:0    138
```

OPENDATAFILE checks to be sure performance item file exists.

```
13    1:D     1 (*$P*)SEGMENT PROCEDURE DEFINEASPECTS;
13    1:0     0   BEGIN
13    1:1     0     ASPECT[1]:='Potentialities';
13    1:1    30     ASPECT[2]:='Processes';
13    1:1    55     ASPECT[3]:='Products';
13    1:1    79     ASPECT[4]:='Environment';
13    1:1   106     ASPECT[5]:='Constraints';
13    1:0   133     END;
13    1:0   146
```

DEFINEASPECTS tells the computer the labels for the aspects.

```
14    1:D     1 (*$P*)SEGMENT PROCEDURE READCOREFILE;
14    1:0     0   BEGIN
14    1:0     0   (*$I-*)
14    1:1     0   RESET(COREFILE,CORENAME);
14    1:1    11   I:=IORESULT;
14    1:1    16   (*$I+*)
14    1:1    16   IF I<>0 THEN
14    1:2    23     BEGIN
14    1:3    23       IF I=9 THEN
14    1:4    30         BEGIN
14    1:5    30           PAGE(OUTPUT);
14    1:5    40           WRITELN('THE APMDISK IS NOT MOUNTED');
14    1:5    86           WRITELN('');
14    1:5   106           WRITELN('PLEASE PLACE IT IN DRIVE #2');
14    1:5   153           ANYKEY;
14    1:5   156           READCOREFILE;
14    1:5   158           EXIT(READCOREFILE)
14    1:4   162           END
14    1:3   162         ELSE
14    1:4   164           BEGIN
14    1:5   164             WRITELN('COREFILE DOES NOT EXIST');
14    1:5   207             WRITELN('  ****FATAL ERROR****  ');
14    1:5   250             WRITELN('             ',I);
14    1:5   293             ANYKEY;
14    1:5   296             BRANCHOUT;
14    1:5   299             SETCHAIN('GREETING');
14    1:5   313             EXIT(PROGRAM);
14    1:4   317             END;
14    1:2   317           END
14    1:1   317       ELSE
14    1:2   319         FOR I:=1 TO 300 DO
14    1:3   335           BEGIN
14    1:4   335             GET(COREFILE);
14    1:4   343             CORE[I]:=COREFILE^;
14    1:3   371             END;
14    1:1   381         GET(COREFILE);
14    1:1   389         CORELAST:=COREFILE^;
14    1:1   405         NCORELAST:=TRUNC(CORELAST);
14    1:1   418         CLOSE(COREFILE)
14    1:0   427       END;
14    1:0   448
```

READCOREFILE reads index to performance items into core.

```
15   1:D     1 (*$P*)SEGMENT PROCEDURE SORTATTRFILE;
15   1:0     0   BEGIN
15   1:1     0     IF NATTRLAST<2 THEN
15   1:2     7       EXIT(SORTATTRFILE);
15   1:1    11     FOR I:=1 TO NATTRIBUTES DO
15   1:2    27       ATTR2[I]:=I;
15   1:1    56     IF NATTRLAST<2 THEN
15   1:2    63       EXIT(SORTATTRFILE);
15   1:1    67     I:=2;
15   1:1    71     REPEAT
15   1:2    71       IF ATTRCORE[I]<ATTRCORE[I-1] THEN
15   1:3   116         BEGIN
15   1:4   116           TEMPX:=ATTRCORE[I];
15   1:4   144           ATTRCORE[I]:=ATTRCORE[I-1];
15   1:4   186           ATTRCORE[I-1]:=TEMPX;
15   1:4   216           TEMP2:=ATTR2[I];
15   1:4   235           ATTR2[I]:=ATTR2[I-1];
15   1:4   269           ATTR2[I-1]:=TEMP2;
15   1:4   290           IF I>2 THEN
15   1:5   297             I:=I-1;
15   1:3   305           END
15   1:2   305         ELSE
15   1:3   307           I:=I+1;
15   1:1   315       UNTIL I>NATTRLAST;
15   1:0   324     END;
15   1:0   342
```

SORTATTRFILE forms an array ATTR2 which is a permutation vector for the attributes file so that if one were to print out ATTRIBUTES [ATTR2(1)] for I=1 to NATTRIBUTES, the attributes would appear in numerical order.

```
16    1:D     1 (*$P*)SEGMENT PROCEDURE SORTMEASFILE;
16    1:0     0   BEGIN
16    1:1     0     IF NMEASLAST<2 THEN
16    1:2     7       EXIT(SORTMEASFILE);
16    1:1    11     FOR I:=1 TO NMEASURES DO
16    1:2    27       MEAS2[I]:=I;
16    1:1    56     IF NMEASLAST<2 THEN
16    1:2    63       EXIT(SORTMEASFILE);
16    1:1    67     I:=2;
16    1:1    71     REPEAT
16    1:2    71       IF MEASCORE[I]<MEASCORE[I-1] THEN
16    1:3   116         BEGIN
16    1:4   116           TEMPX:=MEASCORE[I];
16    1:4   144           MEASCORE[I]:=MEASCORE[I-1];
16    1:4   186           MEASCORE[I-1]:=TEMPX;
16    1:4   216           TEMP2:=MEAS2[I];
16    1:4   235           MEAS2[I]:=MEAS2[I-1];
16    1:4   269           MEAS2[I-1]:=TEMP2;
16    1:4   290           IF I>2 THEN
16    1:5   297             I:=I-1;
16    1:3   305           END
16    1:2   305         ELSE
16    1:3   307           I:=I+1;
16    1:1   315       UNTIL I>NMEASLAST;
16    1:0   324     END;
16    1:0   342
```

SORTMEASFILE forms an array MEAS2 which is a permutation vector for the measures file.

```
17    1:D      1 (*$P*)SEGMENT PROCEDURE SORTCOREFILE;
17    1:0      0   BEGIN
17    1:1      0     FOR I:=1 TO 300 DO
17    1:2     16       CORE2[I]:=I;
17    1:1     45     IF NCORELAST<2 THEN
17    1:2     52       EXIT(SORTCOREFILE);
17    1:1     56     I:=2;
17    1:1     60     REPEAT
17    1:2     60       IF CORE[I]<CORE[I-1] THEN
17    1:3    105         BEGIN
17    1:4    105           TEMP:=CORE[I];
17    1:4    133           CORE[I]:=CORE[I-1];
17    1:4    175           CORE[I-1]:=TEMP;
17    1:4    205           TEMP2:=CORE2[I];
17    1:4    224           CORE2[I]:=CORE2[I-1];
17    1:4    258           CORE2[I-1]:=TEMP2;
17    1:4    279           IF I>2 THEN
17    1:5    286             I:=I-1;
17    1:3    294           END
17    1:2    294         ELSE
17    1:3    296           I:=I+1;
17    1:1    304       UNTIL I>NCORELAST;
17    1:0    313     END;
17    1:0    332
```

SORTCOREFILE prepares an array CORE2 which lists the location of each performance item in numeric order.

```
18 1:D   1 (*$P*)SEGMENT PROCEDURE NAMEFILES;
18 1:0   0    BEGIN
18 1:1   0       APMDSK:=CONCAT(COPY(CURSYS,1,2),COPY(CURSP,1,2),COPY(CURSUB,1,2),':');
18 1:1  86       NAMEATCORE:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'AC');
18 1:1 182       NAMEATTRIBUTES:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,
                 1,4)),'AT');
18 1:1 278       NAMEMECORE:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'MC');
18 1:1 374       NAMEMEASURES:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,
                 1,4)),'ME');
18 1:1 470       CORENAME:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'CO');
18 1:1 566       DATANAME:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'FI');
18 1:1 662       ISSUENAME:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'IS');
18 1:1 758       NAMEFASTISSUE:=CONCAT(APMDSK,COPY(CURSYS,1,4),COPY(CURSP,1,4),COPY(CURSUB,1,4),'FA');
18 1:0 854       END;
18 1:0 866
```

NAMEFILES constructs strings containing the names of files used in this program.

```
19    1:D     1 (*$P*)SEGMENT PROCEDURE DISPLAYNAME;
19    1:0     0   BEGIN
19    1:1     0     SEEK(ISSUE,I);
19    1:1    11     GET(ISSUE);
19    1:1    19     WRITE(I,'. ');
19    1:1    46     FOR J:=1 TO 2 DO
19    1:2    60       BEGIN
19    1:3    60         IF LENGTH(ISSUE^.NAME[J])>60 THEN
19    1:4    81           LINE:=COPY(ISSUE^.NAME[J],1,60)
19    1:3   108           ELSE
19    1:4   112             LINE:=ISSUE^.NAME[J];
19    1:3   132           IF J=2 THEN
19    1:4   139             WRITE('  ');
19    1:3   153           WRITELN(LINE);
19    1:2   173         END;
19    1:1   183     WRITELN(' ');
19    1:0   201   END;
19    1:0   216
```

DISPLAYNAME displays a measurement purpose on screen [called by display issues].

```
20 1:D   1(*$P*)SEGMENT PROCEDURE DISPLAYISSUES;
20 1:0   0  BEGIN
20 1:0   0    (*$I-*)
20 1:1   0    RESET(ISSUE,ISSUENAME);
20 1:1  11    (*$I+*)
20 1:1  11    PAGE(OUTPUT);
20 1:1  21    IF IORESULT<>0 THEN
20 1:2  27      BEGIN
20 1:3  27        NUISSUES:=0;
20 1:3  31        EXIT(DISPLAYISSUES);
20 1:2  35        END;
20 1:1  35    IF NUISSUES=0 THEN
20 1:2  42      BEGIN
20 1:3  42        WRITELN('Currently, there are no measurement issues in the APM for this
                  system and subsystem');
20 1:2 145        END
20 1:1 145      ELSE
20 1:2 147        BEGIN
20 1:3 147          WRITELN('The following measurement purposes are currently included in the APM:');
20 1:3 237          FOR I:=1 TO NUISSUES DO
20 1:4 253            BEGIN
20 1:5 253              DISPLAYNAME;
20 1:5 256              IF (I MOD 6=0) THEN
20 1:6 265                BEGIN
20 1:7 265                  ANYKEY;
20 1:7 268                  PAGE(OUTPUT);
20 1:6 278                  END;
20 1:4 278              END;
20 1:2 288        END;
20 1:1 288    CLOSE(ISSUE);
20 1:0 297    END;
20 1:0 314
20 1:0 314(*$I #5:UTILITY.TEXT*)
20 1:0 314
```

DISPLAYISSUES displays names of all measurement purposes on screen.

```
1    2:D     1 (*$P*)PROCEDURE ANYKEY;
1    2:0     0   BEGIN
1    2:1     0     WRITELN(' ');
1    2:1    18     WRITELN('*** Please press any key to continue ***');
1    2:1    78     (*$R-*)
1    2:1    78     READ(ANS);
1    2:1    89     (*$R+*)
1    2:0    89     END;
1    2:0   102
```

ANYKEY displays "Please Press any Key to Continue" then it awaits a Keypress before returning control to the calling procedure.

```
 1    6:D     1 (*$P*)PROCEDURE HELPER;
 1    6:0     0   BEGIN
 1    6:1     0     WRITELN('For help please refer to your APM MANUAL.');
 1    6:0    61     END;
 1    6:0    74
```

HELPER due to core limitations, it was not possible to implement the full HELP facility. Thus, this HELPER merely displays the message.

```
1   7:D     1 (*$P*)PROCEDURE KEYN;
1   7:D     1   VAR
1   7:D     1     ANSWER: STRING[40];
1   7:D    22     II: ARRAY[1..4] OF INTEGER;
1   7:D    26     OK:BOOLEAN;
1   7:D    27      II0:INTEGER;
1   7:D    28
1   7:0     0   BEGIN
1   7:0     0     (*$R-*)
1   7:1     0     REPEAT
1   7:2     0       REPEAT
1   7:3     0         ANSWER:='                         ';
1   7:3    27         OK:=TRUE;
1   7:3    30         READLN(ANSWER);
1   7:3    49         IF LENGTH(ANSWER)=0 THEN
1   7:4    57           WRITELN('Please enter the integer again');
1   7:2   107         UNTIL LENGTH(ANSWER)<>0;
1   7:2   115       IF (ANSWER[1]='H') OR (ANSWER[1]='h') THEN
1   7:3   130         HELPER;
1   7:2   132       FOR I:=1 TO 4 DO
1   7:3   147         BEGIN
1   7:4   147           II[I]:=ORD(ANSWER[I])-48;
1   7:4   165           IF (II[I]<0) OR (II[I]>9) THEN
1   7:5   192             BEGIN
1   7:6   192               IF (I=1) OR (II[I]<>(ORD(' ')-48)) THEN
1   7:7   214                 BEGIN
1   7:8   214                   OK:=FALSE;
1   7:8   217                   WRITELN('PLEASE RESPOND WITH A POSITIVE INTEGER');
1   7:7   275                   END;
1   7:5   275               END;
1   7:3   275           END;
1   7:1   285       UNTIL OK=TRUE;
1   7:1   292     II0:=II[1];
1   7:1   302     FOR I:=2 TO 4 DO
1   7:2   317       BEGIN
1   7:3   317         IF (II[I]>=0) AND (II[I]<=9) THEN
1   7:4   344           II0:=II0*10+II[I];
1   7:2   361         END;
1   7:2   371     (*$R+*)
1   7:1   371     I:=II0;
1   7:0   376     END;
1   7:0   398
```

KEYN reads a 3 or 4 digit response from the keyboard and places it into I. If an H or an h are typed in, it places a 999 in I and calls the HELP routine: If more than 4 characters are typed, only 4 characters are read. The rest are ignored. If the character(s) are not positive intergers, KEYN will display an appropriate warning and wait for a response.

```
488   1    0:D     1 (*$P*)PROCEDURE KEY;
489   1    0:D     1   VAR
490   1    0:D     1     II2:INTEGER;
491   1    0:0     0   BEGIN
492   1    0:0     0     (*$R-*)
493   1    0:1     0     ANSWER:='                    ';
494   1    0:1    27     REPEAT
495   1    0:2    27       READLN(ANSWER);
496   1    0:2    47       ANS:=ANSWER[1];
497   1    0:2    55       IF (ANS<>'Y') AND (ANS<>'N') AND (ANS<>'H') AND (ANS<>'y') and
498   1    0:2    78         (ANS<>'n') AND (ANS<>'h') AND (ORD(ANS)<>27)THEN
499   1    0:3    98           WRITELN('PLEASE RESPOND YES OR NO!');
500   1    0:2   143       IF (ORD(ANS)>90) THEN
501   1    0:3   150         BEGIN
502   1    0:4   150           II2:=ORD(ANS)-32;
503   1    0:4   157           ANS:=CHR(II2);
504   1    0:3   161           END;
505   1    0:1   161       UNTIL (ANS='Y') OR (ANS='N') OR (ANS='H') OR (ORD(ANS)=27);
506   1    0:1   186       (*$R+*)
507   1    0:1   186       IF ANS='H' THEN
508   1    0:2   193         HELPER;
509   1    0:0   195       END;
510   1    0:0   210
```

KEY reads a letter response from the keyboard. If response is 1) y or Y, it places a Y in ANS and returns to calling procedure; 2) n or N, it places an N in ANS and returns to calling procedure; 3) h or H, it calls the HELP routine, places an H in ANS and returns to calling program; or 4) any other key—it displays PLEASE RESPOND YES OR NO and awaits a Y, N, H, y, n or h response. NOTE: Only the first character/line is processed. The rest is ignored.

```
  1     9:D      1 (*$P*)PROCEDURE PREPKEY(HLP:INTEGER;MSG:STRING);
  1     9:0      0   BEGIN
  1     9:1      0     HELP:=HLP;
  1     9:1      9     REPEAT
  1     9:2      9       WRITE(MSG);
  1     9:2     20       KEY;
  1     9:1     22       UNTIL (ANS='Y') OR (ANS='N') OR (ORD(ANS)=27);
  1     9:0     41     END;
  1     9:0     56
```

PREPKEY displays a message then calls KEY to read a letter response from the keyboard. If a response is not Y, y, N, n, Yes or No, it redisplays the message and, once again, waits for a response.

```
1   10:D     1 (*$P*)PROCEDURE INLINE;
1   10:D     1   VAR
1   10:D     1     LONGLINE:STRING[125];
1   10:D    64     LINEOK:BOOLEAN;
1   10:D    65
1   10:0     0   BEGIN
1   10:1     0     REPEAT
1   10:2     0       READLN(LONGLINE);
1   10:2    19       LINEOK:=TRUE;
1   10:2    22       M:=LENGTH(LONGLINE);
1   10:2    29       IF M>80 THEN
1   10:3    36         BEGIN
1   10:4    36           WRITELN('**WARNING LINE CONTAINS OVER 80 CHARACTERS**');
1   10:4   100           WRITELN(' ');
1   10:4   118           PREPKEY(39,'DO YOU WISH TO TRUNCATE TO 80 CHARACTERS? ');
1   10:4   166           IF ANS='N' THEN
1   10:5   173             BEGIN
1   10:6   173               LINEOK:=FALSE;
1   10:6   176               WRITELN('PLEASE TYPE LINE AGAIN: ');
1   10:5   220               END
1   10:4   220             ELSE
1   10:5   222               M:=80;
1   10:3   226           END;
1   10:1   226       UNTIL LINEOK;
1   10:1   230     LINER:=COPY(LONGLINE,1,M);
1   10:0   248     END;
1   10:0   264
```

INLINE accepts up to 80 characters of text. If more than 80 characters are specified, it asks if it ought to ignore additional characters. If told to, it does. Otherwise, it allows analyst to re-enter the line.

```
1    3:D     1 (*$P*)PROCEDURE BRANCHIN;
1    3:0     0   BEGIN
1    3:0     0     (*$I-*)
1    3:1     0     RESET(PASSNODE,'PASSTHRU');
1    3:1    18     I:=IORESULT;
1    3:1    23     (*$I+*)
1    3:1    23     IF I<>0 THEN
1    3:2    30       BEGIN
1    3:3    30         WRITELN('PASSTHRU FILE DOES NOT EXIST');
1    3:3    78         WRITELN('  ******FATAL ERROR******');
1    3:3   123         WRITELN('              ',I);
1    3:3   167         ANYKEY;
1    3:3   169         SETCHAIN('PGM1');
1    3:3   179         EXIT(PROGRAM);
1    3:2   183         END;
1    3:1   183     GET(PASSNODE);
1    3:1   190     CURSYS:=PASSNODE^.CURSYS;
1    3:1   198     CURSP:=PASSNODE^.CURSP;
1    3:1   206     CURSUB:=PASSNODE^.CURSUB;
1    3:1   214     PAC:=PASSNODE^.PAC;
1    3:1   220     NCURSYS:=PASSNODE^.NCURSYS;
1    3:1   227     NCURSP:=PASSNODE^.NCURSP;
1    3:1   234     NCURSUB:=PASSNODE^.NCURSUB;
1    3:1   241     NPAC:=PASSNODE^.NPAC;
1    3:1   248     CLOSE(PASSNODE);
1    3:0   256     END;
1    3:0   270
```

BRANCHIN gets information from the PASSTHRU file for use by this program.

```
1    4:D     1   (*$P*)PROCEDURE BRANCHOUT;
1    4:0     0     BEGIN
1    4:1     0       REWRITE(PASSNODE,'PASSTHRU');
1    4:1    20       PASSNODE^.FLAG1:=1;
1    4:1    26       PUT(PASSNODE);
1    4:1    33       CLOSE(PASSNODE,LOCK);
1    4:0    41       END;
1    4:0    54
1    4:0    54
1    4:0    54
1    4:0    54 (*$I #5:UTILITY.TEXT*)
1    4:0    54
1    4:0    54 (*$I #5:PRINT2.TEXT*)
```

BRANCHOUT loads the PASSTHRU file with appropriate data for use by called programs.

```
1   11:D     1 (*$P*)PROCEDURE TOPSCREEN;
1   11:0     0      BEGIN
1   11:1     0        PAGE(OUTPUT);
1   11:1    10        M:=LENGTH(CURSYS);
1   11:1    18        IF M>16 THEN
1   11:2    25          M:=16;
1   11:1    29        LINE:=COPY(CURSYS,1,M);
1   11:1    48        WRITE('$',LINE,' Systems');
1   11:1    90        GOTOXY(26,0);
1   11:1    95        M:=LENGTH(CURSP);
1   11:1   103        IF M>16 THEN
1   11:2   110          M:=16;
1   11:1   114        LINE:=COPY(CURSP,1,M);
1   11:1   133        WRITE('$',LINE);
1   11:1   155        GOTOXY(44,0);
1   11:1   160        M:=LENGTH(CURSUB);
1   11:1   168        IF M>16 THEN
1   11:2   175          M:=16;
1   11:1   179        LINE:=COPY(CURSUB,1,M);
1   11:1   198        WRITELN('$',LINE);
1   11:1   228        GOTOXY(62,0);
1   11:1   233        WRITELN('$',PAC);
1   11:1   263        M:=LENGTH(XOBJECTIVE);
1   11:1   271        IF M>67 THEN M:=67;
1   11:1   282        LINE:=COPY(XOBJECTIVE,1,M);
1   11:1   301        IF NSCREEN>1 THEN
1   11:2   308          WRITELN('Objective[',NOBJECTIVE,']:',LINE);
1   11:1   376        M:=LENGTH(XFUNPUR);
1   11:1   384        IF M>67 THEN M:=67;
1   11:1   395        LINE:=COPY(XFUNPUR,1,M);
1   11:1   414        IF NSCREEN>2 THEN
1   11:2   421          WRITELN('Fctl Prps[',NFUNPUR,']:',LINE);
1   11:1   489        WRITELN(' ');
1   11:0   507        END;
1   11:0   520
```

TOPSCREEN displays appropriate header information at the top of each screen.

```
1  12:D    1 (*$P*)PROCEDURE WHICHELIMINATE;
1  12:0    0   BEGIN
1  12:1    0     PAGE(OUTPUT);
1  12:1   10     WRITELN('Would you like to eliminate at the level of:',chr(13),
1  12:1   76             ' 0. No elimination',chr(13),
1  12:1  117             ' 1. Objectives',chr(13),
1  12:1  154             ' 2. Functional Purposes',chr(13),
1  12:1  200             ' 3. Characteristics');
1  12:1  240     REPEAT
1  12:2  240       KEYN;
1  12:2  242       IF (I<0) OR (I>3) OR (I<0) THEN
1  12:3  261         WRITELN('PLEASE SPECIFY AN INTEGER BETWEEN 0 AND 3');
1  12:1  322       UNTIL (I>=0) OR (I<4);
1  12:1  335     IF I=0 THEN
1  12:2  342       BEGIN
1  12:3  342         CLOSE(DATANODE);
1  12:3  351         EXIT(ELIMINATE);
1  12:2  355         END;
1  12:1  355     IF I=1 THEN
1  12:2  362       BEGIN
1  12:3  362         LEVEL:='OBJ';
1  12:3  373         CUT:=10000;
1  12:3  379         NSCREEN:=1;
1  12:2  383         END;
1  12:1  383     IF I=2 THEN
1  12:2  390       BEGIN
1  12:3  390         LEVEL:='FP';
1  12:3  400         CUT:=100;
1  12:3  404         NSCREEN:=2;
1  12:2  408         END;
1  12:1  408     IF I=3 THEN
1  12:2  415       BEGIN
1  12:3  415         LEVEL:='CHAR';
1  12:3  427         CUT:=1;
1  12:3  431         NSCREEN:=3;
1  12:2  435         END;
1  12:0  435     END;
1  12:0  450
```

WHICHELIMINATE asks what level should be used in asking analyst what performance items are <u>not</u> part of his/her analysis.

```
1   13:D     1 (*$P*)PROCEDURE ASKELIMINATE;
1   13:0     0   BEGIN
1   13:1     0     J:=TRUNC(CORE[NODE] DIV 1000000);
1   13:1    54     PAC:=ASPECT[J];
1   13:1    72     NPAC:=J;
1   13:1    78     TOPSCREEN;
1   13:1    80     WRITELN('The following taxon is scheduled to be printed: ');
1   13:1   148     SEEK(DATANODE,CORE2[NODE]);
1   13:1   172     GET(DATANODE);
1   13:1   180     GOTOXY(0,12);
1   13:1   185     WRITE(CHR(11),'  ');
1   13:1   209     FOR J:=1 TO 4 DO
1   13:2   223       WRITE(DATANODE^.NTAXA[J],'.');
1   13:1   266     WRITELN(DATANODE^.TAXA);
1   13:1   288     WRITELN(' ');
1   13:1   306     PREPKEY(230,'Would you like to print it?');
1   13:1   341     IF ORD(ANS)=27 THEN
1   13:2   348       BEGIN
1   13:3   348         CLOSE(DATANODE);
1   13:3   357         EXIT(ELIMINATE);
1   13:2   361         END;
1   13:1   361     IF ANS='Y' THEN
1   13:2   368       EXIT(ASKELIMINATE);
1   13:1   372     PRINTIT[NODE]:=FALSE;
1   13:1   390     IF LEVEL<>'CHAR' THEN
1   13:2   404       FOR I:=1 TO NCORELAST DO
1   13:3   420         IF CORE[I] DIV CUT * CUT = CORE[NODE]
1   13:3   475           THEN PRINTIT[I]:=FALSE;
1   13:0   513     END;
1   13:0   530
```

ASKELIMINATE asks analyst exactly what he/she wants to eliminate by presenting taxons one at a time.

```
 1 5:D    1 (*$P*)PROCEDURE ELIMINATE;
 1 5:0    0   BEGIN
 1 5:1    0     PREPKEY(260,'Do you wish to eliminate any performance items from your printout?');
 1 5:1   74     IF (ANS='N') OR (ORD(ANS)=27) THEN
 1 5:2   87       EXIT(ELIMINATE);
 1 5:1   91     RESET(DATANODE,DATANAME);
 1 5:1  104     WHICHELIMINATE;
 1 5:1  106     FOR NODE:=1 TO NCORELAST DO
 1 5:2  122       IF CORE[NODE]<>0 THEN
 1 5:3  152       BEGIN
 1 5:4  152         IF LEVEL='OBJ' THEN
 1 5:5  165           IF PRINTIT[NODE]=TRUE THEN
 1 5:6  187             IF CORE[NODE]=CORE[NODE] DIV 10000 * 10000 THEN
 1 5:7  252               ASKELIMINATE;
 1 5:4  254         IF LEVEL='FP' THEN
 1 5:5  266           IF PRINTIT[NODE]=TRUE THEN
 1 5:6  288             IF CORE[NODE]=CORE[NODE] DIV 100 * 100 THEN
 1 5:7  349               IF CORE[NODE]<>CORE[NODE] DIV 10000 * 10000 THEN
 1 5:8  414                 ASKELIMINATE;
 1 5:4  416         IF LEVEL='CHAR' THEN
 1 5:5  430           IF PRINTIT[NODE]=TRUE THEN
 1 5:6  452             IF CORE[NODE]<>CORE[NODE] DIV 100 * 100 THEN
 1 5:7  513               ASKELIMINATE;
 1 5:4  515         IF NSCREEN>1 THEN
 1 5:5  522           IF CORE[NODE] DIV 10000 * 10000=CORE[NODE] THEN
 1 5:6  587             BEGIN
 1 5:7  587               SEEK(DATANODE,CORE2[NODE]);
 1 5:7  611               GET(DATANODE);
 1 5:7  619               XOBJECTIVE:=DATANODE^.TAXA;
 1 5:7  629               NOBJECTIVE:=DATANODE^.NTAXA[2];
 1 5:6  644               END;
 1 5:4  644         IF NSCREEN>2 THEN
 1 5:5  651           IF CORE[NODE] DIV 100 * 100 = CORE[NODE] THEN
 1 5:6  712             BEGIN
 1 5:7  712               SEEK(DATANODE,CORE2[NODE]);
 1 5:7  736               GET(DATANODE);
 1 5:7  744               XFUNPUR:=DATANODE^.TAXA;
 1 5:7  754               NFUNPUR:=DATANODE^.NTAXA[3];
 1 5:6  769               END;
 1 5:3  769         END;
 1 5:1  779     CLOSE(DATANODE);
 1 5:0  788     END;
 1 5:0  810
```

ELIMINATE using information gained from WHICHELIMINATE, calls ASKELIMINATE as appropriate.

```
1 14:D   1   (*$P*)PROCEDURE PROCESSISSUE;
1 14:0   0     BEGIN
1 14:1   0       IF NOISSUE=FALSE THEN
1 14:2   8         BEGIN
1 14:3   8           REPEAT
1 14:4   8             WRITE('Which measurement purpose would you like to use (type 0 for none)?');
1 14:4  86             KEYN;
1 14:3  88             UNTIL (I>=0) AND (I<=NISSUES)
1 14:2 100           END
1 14:1 103         ELSE
1 14:2 105           I:=0;
1 14:1 109       NCURISSUE:=I;
1 14:1 115       IF I=0 THEN
1 14:2 122         BEGIN
1 14:3 122           FOR J:=1 TO 300 DO
1 14:4 138             PRINTIT[J]:=TRUE;
1 14:3 166           EXIT (PROCESSISSUE);
1 14:2 170           END;
1 14:1 170       RESET(FASTISSUE,NAMEFASTISSUE);
1 14:1 183       SEEK(FASTISSUE,I);
1 14:1 194       GET(FASTISSUE);
1 14:1 202       CLOSE(FASTISSUE);
1 14:1 211       OK:=FALSE;
1 14:1 215       FOR J:=1 TO 300 DO
1 14:2 231         BEGIN
1 14:3 231           PRINTIT[J]:=FASTISSUE^.PRINTIT[J];
1 14:3 265           IF PRINTIT[J]=TRUE THEN
1 14:4 287             OK:=TRUE;
1 14:2 291           END;
1 14:1 301       IF OK=TRUE THEN
1 14:2 309         EXIT(PROCESSISSUE);
1 14:1 313       FOR J:=1 TO 300 DO
1 14:2 329         PRINTIT[J]:=FALSE;
1 14:1 357       RESET(ISSUE,ISSUENAME);
1 14:1 370       SEEK(ISSUE,I);
1 14:1 381       GET(ISSUE);
1 14:1 389       T2:=0;
1 14:1 404       WRITELN('Please be patient...',chr(13),
1 14:1 446               ' I am getting set up to use your measurement purpose');
1 14:1 519       FOR J:=1 TO 225 DO
```

PROCESSISSUE selects performance items for printing based upon the measurement purpose in use.

```
772  1  14:2  535        BEGIN
773  1  14:3  535          T1:=ISSUE^.DATA[J];
774  1  14:3  565          IF T1<>0 THEN
775  1  14:4  583            BEGIN
776  1  14:5  583              SKIP:=FALSE;
777  1  14:5  587              FOR K:=1 TO 300 DO
778  1  14:6  603                BEGIN
779  1  14:7  603                  T5:=CORE[K];
780  1  14:7  631                  IF (T1 = T5)THEN
781  1  14:8  650                    BEGIN
782  1  14:9  650                      PRINTIT[K]:=TRUE;
783  1  14:9  668                      SKIP:=TRUE;
784  1  14:8  672                      END;
785  1  14:6  672                  END;
786  1  14:5  682              IF SKIP=TRUE THEN
787  1  14:6  690                IF (T1 DIV 100 * 100 <> T2) THEN
788  1  14:7  727                  BEGIN
789  1  14:8  727                    FOR K:=1 TO 300 DO
790  1  14:9  743                      BEGIN
791  1  14:0  743                        T2:=T1 DIV 100*100;
792  1  14:0  777                        T3:=T1 DIV 10000 * 10000;
793  1  14:0  815                        T4:=T1 DIV 1000000 *1000000;
794  1  14:0  889                        T5:=CORE[K];
795  1  14:0  917                        IF T2 = T5 THEN
796  1  14:1  936                          PRINTIT[K]:=TRUE;
797  1  14:0  954                        IF T3 = T5 THEN
798  1  14:1  973                          PRINTIT[K]:=TRUE;
799  1  14:0  991                        IF T4 = T5 THEN
800  1  14:1  1010                         PRINTIT[K]:=TRUE;
801  1  14:9  1028                       END;
802  1  14:7  1038                  END;
803  1  14:4  1038            END;
804  1  14:2  1038        END;
805  1  14:1  1048     CLOSE(ISSUE);
806  1  14:1  1057     RESET(FASTISSUE,NAMEFASTISSUE);
807  1  14:1  1070     J:=NCURISSUE;
808  1  14:1  1076     SEEK(FASTISSUE,J);
809  1  14:1  1087     FOR J:=1 TO 300 DO
810  1  14:2  1103       FASTISSUE^.PRINTIT[J]:=PRINTIT[J];
811  1  14:1  1147     PUT(FASTISSUE);
812  1  14:1  1155     CLOSE(FASTISSUE);
813  1  14:0  1164     END;
814  1  14:0  1202
```

See previous page for program description.

```
1   15:D     1 (*$P*)PROCEDURE PRINTMEASURE;
1   15:0     0   BEGIN
1   15:1     0     FOR NCURMEASURE:=1 TO NMEASLAST DO
1   15:2    16       IF ATTRCORE[NCURATTRIBUTE]=MEASCORE[NCURMEASURE] DIV 100 THEN
1   15:3    68         BEGIN
1   15:4    68           SEEK(MEASURES,MEAS2[NCURMEASURE]);
1   15:4    92           GET(MEASURES);
1   15:4   100           WRITE(PRNT,'              ');
1   15:4   124           FOR K:=1 TO 6 DO
1   15:5   138             WRITE(PRNT,MEASURES^.NDESCRIPTOR[K],'.');
1   15:4   181           WRITELN(PRNT,' ',MEASURES^.DESCRIPTOR);
1   15:3   213           END;
1   15:0   223     END;
1   15:0   244
```

PRINTMEASURE prints a measure for current performance item.

```
1   16:D      1 (*$P*)PROCEDURE PRINTATTRIBUTE;
1   16:0      0   BEGIN
1   16:1      0     FOR NCURATTRIBUTE:=1 TO NATTRLAST DO
1   16:2     16       IF CORE[I]=ATTRCORE[NCURATTRIBUTE] DIV 100 THEN
1   16:3     68         BEGIN
1   16:4     68           SEEK(ATTRIBUTES,ATTR2[NCURATTRIBUTE]);
1   16:4     92           GET(ATTRIBUTES);
1   16:4    100           WRITE(PRNT,'            ');
1   16:4    122           FOR K:=1 TO 6 DO
1   16:5    136             WRITE(PRNT,ATTRIBUTES^.NDESCRIPTOR[K],'.');
1   16:4    179           WRITELN(PRNT,' ',ATTRIBUTES^.DESCRIPTOR);
1   16:4    211           PRINTMEASURE;
1   16:3    213           END;
1   16:0    223     END;
1   16:0    244
```

PRINTATTRIBUTE prints attributes for current performance items.

```
844 1 17:D   1(*$P*)PROCEDURE GETUSERSTUFF;
845 1 17:0   0  BEGIN
846 1 17:1   0    USERNAME:='$$$$$$$$$$';
847 1 17:1  18    USERMSG:='$$$$$$$$$$';
848 1 17:1  36    REPEAT
849 1 17:2  36      WRITE('What is your name? ');
850 1 17:2  67      (*$R-*) .
851 1 17:2  67      READLN(USERNAME);
852 1 17:2  87      (*$R+*)
853 1 17:2  87      WRITELN(' ');
854 1 17:1 105      UNTIL (COPY(USERNAME,1,3)<>'$$$') AND (LENGTH(USERNAME)>0);
855 1 17:1 135    REPEAT
856 1 17:2 135      WRITELN('Please type a 40-character (max.) identification code for the printout!');
857 1 17:2 226      (*$R-*)
858 1 17:2 226      READLN(USERMSG);
859 1 17:2 246      (*$R+*)
860 1 17:2 246      WRITELN(' ');
861 1 17:1 264      UNTIL (COPY(USERMSG,1,3)<>'$$$') AND (LENGTH(USERMSG)>0);
862 1 17:1 294    REPEAT
863 1 17:2 294      WRITELN('Please type todays date! ');
864 1 17:2 339      (*$R-*)
865 1 17:2 339      READLN(USERDATE);
866 1 17:2 359      (*$R+*)
867 1 17:2 359      WRITELN(' ');
868 1 17:1 377      UNTIL (COPY(USERMSG,1,3)<>'$$$') AND (LENGTH(USERDATE)>0);
869 1 17:0 407    END;
870 1 17:0 426
```

GETUSERSTUFF asks analyst for his/her name, project title and the date.

```
1   18:D     1   (*$P*)PROCEDURE TITLEPAGE;
1   18:0     0     BEGIN
1   18:1     0       REWRITE(PRNT,'PRINTER:');
1   18:1    21       FOR I:=1 TO 13 DO
1   18:2    35         WRITELN(PRNT,CHR(14),' ');
1   18:1    73       WRITELN(PRNT,CHR(14),'                    An Analytic Process Model For');
1   18:1   147       WRITELN(PRNT,CHR(14),' ');
1   18:1   175       WRITELN(PRNT,CHR(14),'                  Systems Design And Measurement:');
1   18:1   250       FOR I:=1 TO 5 DO
1   18:2   264         WRITELN(PRNT,CHR(14),' ');
1   18:1   302       WRITELN(PRNT,CHR(14),'                  Listing Of Taxa and Measurements');
1   18:1   378       FOR I:=1 TO 5 DO
1   18:2   392         WRITELN(PRNT,CHR(14),' ');
1   18:1   430       WRITELN(PRNT,CHR(14),'For: ',USERNAME);
1   18:1   477       WRITELN(PRNT,CHR(14),' ');
1   18:1   505       WRITELN(PRNT,CHR(14),'Date: ',USERDATE);
1   18:1   553       WRITELN(PRNT,CHR(14),' ');
1   18:1   581       WRITELN(PRNT,CHR(14),'Re: ',USERMSG);
1   18:1   627       CLOSE(PRNT);
1   18:0   636       END;
1   18:0   654
```

TITLEPAGE prints title page for printout.

```
1   19:D     1  {$$P$}PROCEDURE HEADER;
1   19:0     0    BEGIN
1   19:1     0      REWRITE(PRNT,'PRINTER:');
1   19:1    21      PAGE(PRNT);
1   19:1    31      WRITELN(PRNT,CHR(14),USERNAME);
1   19:1    61      WRITELN(PRNT,CHR(14),USERDATE);
1   19:1    91      WRITELN(PRNT,CHR(14),USERMSG);
1   19:1   121      IF NCURISSUE<>0 THEN
1   19:2   128        BEGIN
1   19:3   128          RESET(ISSUE,ISSUENAME);
1   19:3   141          SEEK(ISSUE,NCURISSUE);
1   19:3   152          GET(ISSUE);
1   19:3   160          WRITELN(PRNT,' ');
1   19:3   178          WRITELN(PRNT,CHR(14),'Measurement Purpose: ',CHR(15),ISSUE^.NAME[1]);
1   19:3   261          WRITELN(PRNT,'                              ',ISSUE^.NAME[2]);
1   19:3   328          WRITELN(PRNT,' ');
1   19:3   346          CLOSE(ISSUE);
1   19:2   355          END
1   19:1   355        ELSE
1   19:2   357          BEGIN
1   19:3   357            WRITELN(PRNT,' ');
1   19:3   375            WRITELN(PRNT,CHR(14),'Measurement Purpose: ',CHR(15),' Global');
1   19:3   455            WRITELN(PRNT,' ');
1   19:2   473            END;
1   19:1   473      WRITELN(PRNT,CHR(14),'System Class: ',chr(15),CURSYS,'[',NCURSYS,']');
1   19:1   571      WRITELN(PRNT,CHR(14),'System: ',chr(15),cursp,'[',NCURSP,']');
1   19:1   663      WRITELN(PRNT,CHR(14),'Subsystem: ',chr(15),CURSUB,'[',NCURSUB,']');
1   19:1   758      WRITELN(PRNT,' ',CHR(15),CHR(13));
1   19:0   796      END;
1   19:0   810
```

HEADER prints header on printout.

```
1   20:D     1   (*$P*)PROCEDURE PRNTDATASET;
1   20:0     0     BEGIN
1   20:1     0       RESET(DATANODE,DATANAME);
1   20:1    13       RESET(ATTRIBUTES,NAMEATTRIBUTES);
1   20:1    26       RESET(MEASURES,NAMEMEASURES);
1   20:1    39       REPEAT
1   20:2    39         HEADER;
1   20:2    41         TEMP2:=0;
1   20:2    45         FOR I:=1 TO NCORELAST DO
1   20:3    61           IF PRINTIT[I]=TRUE THEN
1   20:4    83             BEGIN
1   20:5    83               SEEK(DATANODE,CORE2[I]);
1   20:5   107               GET(DATANODE);
1   20:5   115               INDENT:=4;
1   20:5   119               IF DATANODE^.NTAXA[4]=0 THEN
1   20:6   135                 INDENT:=3;
1   20:5   139               IF DATANODE^.NTAXA[3]=0 THEN
1   20:6   155                 INDENT:=2;
1   20:5   159               IF DATANODE^.NTAXA[2]=0 THEN
1   20:6   175                 INDENT:=1;
1   20:5   179               IF (DATANODE^.NTAXA[1]<>TEMP2) AND (DATANODE^.NTAXA[1]<>0) THEN
1   20:6   212                 BEGIN
1   20:7   212                   WRITE(PRNT,DATANODE^.NTAXA[1],'.0.0.0.0.0. ');
1   20:7   257                   WRITE(PRNT,ASPECT[DATANODE^.NTAXA[1]],': ');
1   20:7   302                   CASE DATANODE^.NTAXA[1] OF
1   20:7   316                     1:WRITELN(PRNT,OBJLBL1);
1   20:7   368                     2:WRITELN(PRNT,OBJLBL2);
1   20:7   441                     3:WRITELN(PRNT,OBJLBL3);
1   20:7   487                     4:WRITELN(PRNT,OBJLBL4);
1   20:7   552                     5:WRITELN(PRNT,OBJLBL5);
1   20:7   617                     END;
1   20:6   634                   END;
1   20:5   634               TEMP2:=DATANODE^.NTAXA[1];
1   20:5   649               FOR J:=1 TO INDENT DO
1   20:6   665                 WRITE(PRNT,'  ');
1   20:5   689               FOR J:=1 TO 4 DO
1   20:6   703                 WRITE(PRNT,DATANODE^.NTAXA[J],'.');
1   20:5   746               WRITELN(PRNT,'0.0. ',DATANODE^.TAXA);
1   20:5   785               IF INDENT>1 THEN
1   20:6   792                 PRINTATTRIBUTE;
```

PRINTDATASET is the controlling program for printing a data set. Also, prints all performance items. Calls PRINTATTRIBUTE when necessary.

```
962 1 20:5  794              IF KEYPRESS THEN
963 1 20:6  801                BEGIN
964 1 20:7  801                  READ(ANS);
965 1 20:7  812                  IF ORD(ANS)=27 THEN
966 1 20:8  819                    BEGIN
967 1 20:9  819                      WRITELN(PRNT,'Job cancelled');
968 1 20:9  852                      PAGE(PRNT);
969 1 20:9  862                      CLOSE(PRNT);
970 1 20:9  871                      CLOSE(DATANODE);
971 1 20:9  880                      CLOSE(ATTRIBUTES);
972 1 20:9  889                      CLOSE(MEASURES);
973 1 20:9  898                      EXIT(PRNTDATASET);
974 1 20:8  902                      END;
975 1 20:6  902                   END;
976 1 20:4  902              END;
977 1 20:2  912            PAGE(PRNT);
978 1 20:2  922            CLOSE(PRNT);
979 1 20:2  931            PREPKEY(303,'Would you like to print another copy of these measurements?');
980 1 20:1  998            UNTIL (ANS='N') OR (ORD(ANS)=27);
981 1 20:1 1011          CLOSE(DATANODE);
982 1 20:1 1020          CLOSE(ATTRIBUTES);
983 1 20:1 1029          CLOSE(MEASURES);
984 1 20:0 1038          END;
985 1 20:0 1068
986 1 20:0 1068
987 1 20:0 1068 (*$I #5:PRINT2.TEXT*)
988 1 20:0 1068
```

See previous page for program description.

```
 989 1 1:0   0 (*$P*)BEGIN
 990 1 1:0   0   (*$N-*)
 991 1 1:1   0   NISSUES:=5;
 992 1 1:1 118   NMEASURES:=400;
 993 1 1:1 124   NATTRIBUTES:=200;
 994 1 1:1 130   BRANCHIN;
 995 1 1:1 132   DEFINEASPECTS;
 996 1 1:1 135   NAMEFILES;
 997 1 1:1 138   WRITELN('Please be patient');
 998 1 1:1 175   WRITELN('  I am starting to sort your datafiles');
 999 1 1:1 233   READATTRFILE;
1000 1 1:1 236   SORTATTRFILE;
1001 1 1:1 239   WRITELN('  I just finished sorting the attributes');
1002 1 1:1 299   READMEASFILE;
1003 1 1:1 302   SORTMEASFILE;
1004 1 1:1 305   WRITELN('  I just finished sorting the measures');
1005 1 1:1 363   READCOREFILE;
1006 1 1:1 366   SORTCOREFILE;
1007 1 1:1 369   WRITELN('  I just finished sorting your datafiles');
1008 1 1:1 429   OPENDATAFILE;
1009 1 1:1 432   OPENISSUEINDEX;
1010 1 1:1 435   OPENFASTISSUE;
1011 1 1:1 438   GETUSERSTUFF;
1012 1 1:1 440   TITLEPAGE;
1013 1 1:1 442   REPEAT
1014 1 1:2 442     DISPLAYISSUES;
1015 1 1:2 445     PROCESSISSUE;
1016 1 1:2 447     REPEAT
1017 1 1:3 447       ELIMINATE;
1018 1 1:3 449       PRNTDATASET;
1019 1 1:3 451       PREPKEY(359,'Would you like to remove more performance items from your printout?');
1020 1 1:2 526       UNTIL (ANS='N') OR (ORD(ANS)=27);
1021 1 1:2 539     PREPKEY(360,'Would you like to process another measurement purpose?')
1022 1 1:1 599     UNTIL (ANS='N') OR (ORD(ANS)=27);
1023 1 1:1 614   REWRITE(PRNT,'PRINTER:');
1024 1 1:1 635   FOR I:=1 TO 10 DO
1025 1 1:2 652     WRITELN(PRNT,'END OF PRINTOUT FOR ',USERNAME);
1026 1 1:1 714   PAGE(PRNT);
1027 1 1:1 724   CLOSE(PRNT);
1028 1 1:1 733   BRANCHOUT;
1029 1 1:1 735   SETCHAIN('GREETING');
1030 1 1:0 749   END.
```

BEGIN is the main program: 1) sorts attributes and measures, 2) processes measurement purposes, 3) eliminates unwanted performance items, and 4) prints wanted performance items.

```
962 1 20:5  794             IF KEYPRESS THEN
963 1 20:6  801               BEGIN
964 1 20:7  801                 READ(ANS);
965 1 20:7  812                 IF ORD(ANS)=27 THEN
966 1 20:8  819                   BEGIN
967 1 20:9  819                     WRITELN(PRNT,'Job cancelled');
968 1 20:9  852                     PAGE(PRNT);
969 1 20:9  862                     CLOSE(PRNT);
970 1 20:9  871                     CLOSE(DATANODE);
971 1 20:9  880                     CLOSE(ATTRIBUTES);
972 1 20:9  889                     CLOSE(MEASURES);
973 1 20:9  898                     EXIT(PRNTDATASET);
974 1 20:8  902                     END;
975 1 20:6  902                 END;
976 1 20:4  902             END;
977 1 20:2  912           PAGE(PRNT);
978 1 20:2  922           CLOSE(PRNT);
979 1 20:2  931           PREPKEY(303,'Would you like to print another copy of these measurements?');
980 1 20:1  998           UNTIL (ANS='N') OR (ORD(ANS)=27);
981 1 20:1 1011         CLOSE(DATANODE);
982 1 20:1 1020         CLOSE(ATTRIBUTES);
983 1 20:1 1029         CLOSE(MEASURES);
984 1 20:0 1038         END;
985 1 20:0 1068
986 1 20:0 1068
987 1 20:0 1068 (*$I #5:PRINT2.TEXT*)
988 1 20:0 1068
```

See previous page for program description.

```
1 1:0   0 (*$P*)BEGIN
1 1:0   0   (*$N-*)
1 1:1   0   NISSUES:=5;
1 1:1 118   NMEASURES:=400;
1 1:1 124   NATTRIBUTES:=200;
1 1:1 130   BRANCHIN;
1 1:1 132   DEFINEASPECTS;
1 1:1 135   NAMEFILES;
1 1:1 138   WRITELN('Please be patient');
1 1:1 175   WRITELN('  I am starting to sort your datafiles');
1 1:1 233   READATTRFILE;
1 1:1 236   SORTATTRFILE;
1 1:1 239   WRITELN('  I just finished sorting the attributes');
1 1:1 299   READMEASFILE;
1 1:1 302   SORTMEASFILE;
1 1:1 305   WRITELN('  I just finished sorting the measures');
1 1:1 363   READCOREFILE;
1 1:1 366   SORTCOREFILE;
1 1:1 369   WRITELN('  I just finished sorting your datafiles');
1 1:1 429   OPENDATAFILE;
1 1:1 432   OPENISSUEINDEX;
1 1:1 435   OPENFASTISSUE;
1 1:1 438   GETUSERSTUFF;
1 1:1 440   TITLEPAGE;
1 1:1 442   REPEAT
1 1:2 442     DISPLAYISSUES;
1 1:2 445     PROCESSISSUE;
1 1:2 447     REPEAT
1 1:3 447       ELIMINATE;
1 1:3 449       PRNTDATASET;
1 1:3 451       PREPKEY(359,'Would you like to remove more performance items from your printout?');
1 1:2 526       UNTIL (ANS='N') OR (ORD(ANS)=27);
1 1:2 539     PREPKEY(360,'Would you like to process another measurement purpose?')
1 1:1 599     UNTIL (ANS='N') OR (ORD(ANS)=27);
1 1:1 614   REWRITE(PRNT,'PRINTER:');
1 1:1 635   FOR I:=1 TO 10 DO
1 1:2 652     WRITELN(PRNT,'END OF PRINTOUT FOR ',USERNAME);
1 1:1 714   PAGE(PRNT);
1 1:1 724   CLOSE(PRNT);
1 1:1 733   BRANCHOUT;
1 1:1 735   SETCHAIN('GREETING');
1 1:0 749   END.
```

BEGIN is the main program: 1) sorts attributes and measures, 2) processes measurement purposes, 3) eliminates unwanted performance items, and 4) prints wanted performance items.

PACK

Pack causes the performance item, attribute and measures data sets to be sorted into numerical order (according to statement number). It also moves unused space to the end of each data set where it becomes available for use with subsequent execution of the PERFITEM and MEASATTR programs.

```
 1    1:D     1 (*$L PRINTER: *)
 1    1:D     1 (*$S+*)
 1    1:D     1 (* Program to pack performance items, attribute, and measures lists *)
 1    1:D     1 (* Ronald G. Shapiro   Version 2.0         10/25/82*)
 1    1:D     1 Program Packdatasetts;
 1    1:D     3
28    1:D     3
28    2:D     1   PROCEDURE SETCHAIN(TYTLE:STRING);
28    3:D     1   PROCEDURE SETCVAL(VAL:STRING);
28    4:D     1   PROCEDURE GETCVAL(VAR VAL:STRING);
28    5:D     1   PROCEDURE SWAPON;
28    6:D     1   PROCEDURE SWAPOFF;
28    6:D     1
 1    1:D     1 Uses Chainstuff;
 1    1:D     3
```

These procedures are part of the Apple Computer's CHAINSTUFF library entry. The demonstration package uses only SETCHAIN which causes another program to be activated.

```
1   1:D    3 (*$P*)TYPE
1   1:D    3   ISSUEFILE =RECORD
1   1:D    3     NUM:INTEGER;
1   1:D    3     NAME:ARRAY[1..2]OF STRING[80];
1   1:D    3     DATA:ARRAY[1..225]OF INTEGER[8];
1   1:D    3     END;
1   1:D    3
1   1:D    3   PASSFILE =RECORD
1   1:D    3     CURSYS,CURSP,CURSUB,PAC:STRING[80];
1   1:D    3     NCURSYS,NCURSP,NCURSUB,NPAC,FLAG1,FLAG2,FLAG3:INTEGER;
1   1:D    3     END;
1   1:D    3
1   1:D    3   DATABASE =RECORD
1   1:D    3     NTAXA: ARRAY[1..4] OF INTEGER;
1   1:D    3     TAXA: STRING[80];
1   1:D    3     END;
1   1:D    3
1   1:D    3   FILEATTRIBUTES =RECORD
1   1:D    3     NDESCRIPTOR: ARRAY[1..6] OF INTEGER;
1   1:D    3     DESCRIPTOR: STRING[68];
1   1:D    3     END;
1   1:D    3
1   1:D    3   FILEMEASURES =RECORD
1   1:D    3     NDESCRIPTOR: ARRAY[1..6] OF INTEGER;
1   1:D    3     DESCRIPTOR: STRING[68];
1   1:D    3     END;
1   1:D    3
1   1:D    3   TEMPM =RECORD
1   1:D    3     NDESCRIPTOR: ARRAY[1..6] OF INTEGER;
1   1:D    3     DESCRIPTOR: STRING[68];
1   1:D    3     END;
1   1:D    3
1   1:D    3   TEMPA =RECORD
1   1:D    3     NDESCRIPTOR: ARRAY[1..6] OF INTEGER;
1   1:D    3     DESCRIPTOR: STRING[68];
1   1:D    3     END;
1   1:D    3
1   1:D    3   TEMPD =RECORD
1   1:D    3     NTAXA: ARRAY[1..4] OF INTEGER;
1   1:D    3     TAXA: STRING[80];
1   1:D    3     END;
1   1:D    3
```

ISSUEFILE contains the measurement purposes. PASSFILE passes information between the various programs. DATABASE contains the performance items. FILEATTRIBUTES contains the attributes. FILEMEASURES contains the measures. TEMPM, TEMPA, TEMPD are temporary files used during the pack procedure.

```
58   1    1:D     3 (*$P*)VAR
59   1    1:D     3   PASSNODE:FILE OF PASSFILE;
60   1    1:D   474   DATANODE:FILE OF DATABASE;
61   1    1:D   819   COREFILE:FILE OF INTEGER[8];
62   1    1:D  1122   ATTRIBUTES:FILE OF FILEATTRIBUTES;
63   1    1:D  1463   ATTRFILE:FILE OF INTEGER[12];
64   1    1:D  1767   MEASURES:FILE OF FILEMEASURES;
65   1    1:D  2108   MEASFILE:FILE OF INTEGER[12];
66   1    1:D  2412   ISSUE:FILE OF ISSUEFILE;
67   1    1:D  3470   TEMPMEASURES:FILE OF TEMPM;
68   1    1:D  3811   TEMPATTRIBUTES:FILE OF TEMPA;
69   1    1:D  4152   TEMPDATA:FILE OF TEMPD;
70   1    1:D  4497
71   1    1:D  4497   CORE:ARRAY[1..300] OF INTEGER[8];
72   1    1:D  5397   ATTRCORE:ARRAY[1..200] OF INTEGER[12];
73   1    1:D  6197   MEASCORE:ARRAY[1..400] OF INTEGER[12];
74   1    1:D  7797   SCRATCH:ARRAY [1..20] OF INTEGER;
75   1    1:D  7817   ASPECT:ARRAY[1..5] OF STRING[20];
76   1    1:D  7872   NASPECT:ARRAY[1..5] OF INTEGER;
77   1    1:D  7877   CORE2:ARRAY[1..300] OF INTEGER;
78   1    1:D  8177   ATTR2:ARRAY[1..200] OF INTEGER;
79   1    1:D  8377   MEAS2:ARRAY[1..400] OF INTEGER;
80   1    1:D  8777   PRINTIT:ARRAY[1..300] OF BOOLEAN;
81   1    1:D  9077
82   1    1:D  9077   XCHARAC,XFUNPUR,XOBJECTIVE,PAC,CURSYS,CURSP,CURSUB: STRING[80];
83   1    1:D  9364   NCURMEASURE,NCURATTRIBUTE,NCURISSUE,NCHARAC,
84   1    1:D  9364     NFUNPUR,NOBJECTIVE,NPAC,NCURSYS,NCURSP,NCURSUB: INTEGER;
85   1    1:D  9374
86   1    1:D  9374   ISSUENAME,NAMEATCORE,NAMEATTRIBUTES,NAMEMECORE,NAMEMEASURES: STRING[40];
87   1    1:D  9479   NAMETEMPORARY,CORENAME,DATANAME: STRING[40];
88   1    1:D  9542   APMDSK:STRING[10];
89   1    1:D  9548   LEVEL: STRING[10];
90   1    1:D  9554   USERNAME,USERDATE,USERMSG: STRING[80];
91   1    1:D  9677
92   1    1:D  9677   TEMP,CORELAST,T1,T2,T3,T4,T5: INTEGER[8];
93   1    1:D  9698   TEMPX,ATTRLAST,MEASLAST:INTEGER[12];
94   1    1:D  9710
95   1    1:D  9710   MODE,INVERSE,HELP,NSCREEN:INTEGER;
96   1    1:D  9714   NCORELAST,NATTRLAST,NMEASLAST:INTEGER;
97   1    1:D  9717   NISSUES,NUISSUES,NATTRIBUTES,NMEASURES,NUMEASURES:INTEGER;
```

These strings, arrays and variables are used by this program.

```
1     1:D  9722
1     1:D  9722   I,J,K,L,M,N,CUT,INDENT,COUNT,TEMP2:INTEGER;
1     1:D  9732
1     1:D  9732   REFERENCED,LONGWAY,DONE,OVER,OK,SKIP,NONE:BOOLEAN;
1     1:D  9739
1     1:D  9739   LINER:STRING[80];
1     1:D  9780   LINE:STRING[60];
1     1:D  9811
1     1:D  9811   ANSWER,REGLINE:STRING[80];
1     1:D  9893
1     1:D  9893   ANS,ANSHOLD: CHAR;
1     1:D  9895
1     1:D  9895   PRNT:TEXT;
1     1:D 10196
1     1:D 10196 (*$I #5:UTILITY.TEXT*)
1     1:D 10196
```

See previous page for program description.

```
1    2:D     1 (*$P*)PROCEDURE ANYKEY;
1    2:0     0   BEGIN
1    2:1     0     WRITELN(' ');
1    2:1    18     WRITELN('*** Please press any key to continue ***');
1    2:1    78     (*$R-*)
1    2:1    78     READ(ANS);
1    2:1    89     (*$R+*)
1    2:0    89     END;
1    2:0   102
```

ANYKEY displays "Please Press any Key to Continue" then it awaits a Keypress before returning control to the calling procedure.

```
122   1    3:D     1 (*$P*)PROCEDURE HELPER;
123   1    3:0     0   BEGIN
124   1    3:1     0     WRITELN('For help please refer to your APM MANUAL.');
125   1    3:0    61     END;
126   1    3:0    74
```

HELPER due to core limitations, it was not possible to implement the full HELP facility. Thus, this HELPER merely displays the message.

```
1    4:D     1 (*$P*)PROCEDURE KEYN;
1    4:D     1   VAR
1    4:D     1     ANSWER: STRING[40];
1    4:D    22     II: ARRAY[1..4] OF INTEGER;
1    4:D    26     OK:BOOLEAN;
1    4:D    27      IIO:INTEGER;
1    4:D    28
1    4:0     0   BEGIN
1    4:0     0     (*$R-*)
1    4:1     0     REPEAT
1    4:2     0       REPEAT
1    4:3     0         ANSWER:='                          ';
1    4:3    27         OK:=TRUE;
1    4:3    30         READLN(ANSWER);
1    4:3    49         IF LENGTH(ANSWER)=0 THEN
1    4:4    57           WRITELN('Please enter the integer again');
1    4:2   107         UNTIL LENGTH(ANSWER)<>0;
1    4:2   115       IF (ANSWER[1]='H') OR (ANSWER[1]='h') THEN
1    4:3   130         HELPER;
1    4:2   132       FOR I:=1 TO 4 DO
1    4:3   147         BEGIN
1    4:4   147           II[I]:=ORD(ANSWER[I])-48;
1    4:4   165           IF (II[I]<0) OR (II[I]>9) THEN
1    4:5   192             BEGIN
1    4:6   192               IF (I=1) OR (II[I]<>(ORD(' ')-48)) THEN
1    4:7   214                 BEGIN
1    4:8   214                   OK:=FALSE;
1    4:8   217                   WRITELN('PLEASE RESPOND WITH A POSITIVE INTEGER');
1    4:7   275                   END;
1    4:5   275               END;
1    4:3   275           END;
1    4:1   285       UNTIL OK=TRUE;
1    4:1   292     IIO:=II[1];
1    4:1   302     FOR I:=2 TO 4 DO
1    4:2   317       BEGIN
1    4:3   317         IF (II[I]>=0) AND (II[I]<=9) THEN
1    4:4   344           IIO:=IIO*10+II[I];
1    4:2   361         END;
1    4:2   371     (*$R+*)
1    4:1   371     I:=IIO;
1    4:0   376     END;
1    4:0   398
```

KEYN reads a 3 or 4 digit response from the keyboard and places it into I. If an H or an h are typed in, it places a 999 in I and calls the HELP routine. If more than 4 characters are typed, only 4 characters are read. The rest are ignored. If the character(s) are not positive intergers, KEYN will display an appropriate warning and wait for a response.

```
1   5:D    1 (*$P*)PROCEDURE KEY;
1   5:D    1   VAR
1   5:D    1     II2:INTEGER;
1   5:0    0   BEGIN
1   5:0    0     (*$R-*)
1   5:1    0     ANSWER:='                    ';
1   5:1   27     REPEAT
1   5:2   27       READLN(ANSWER);
1   5:2   47       ANS:=ANSWER[1];
1   5:2   55       IF (ANS<>'Y') AND (ANS<>'N') AND (ANS<>'H') AND (ANS<>'y') and
1   5:2   78         (ANS<>'n') AND (ANS<>'h') AND (ORD(ANS)<>27)THEN
1   5:3   98           WRITELN('PLEASE RESPOND YES OR NO!');
1   5:2  143       IF (ORD(ANS)>90) THEN
1   5:3  150         BEGIN
1   5:4  150           II2:=ORD(ANS)-32;
1   5:4  157           ANS:=CHR(II2);
1   5:3  161           END;
1   5:1  161       UNTIL (ANS='Y') OR (ANS='N') OR (ANS='H') OR (ORD(ANS)=27);
1   5:1  186       (*$R+*)
1   5:1  186       IF ANS='H' THEN
1   5:2  193         HELPER;
1   5:0  195       END;
1   5:0  210
```

KEY reads a letter response from the keyboard. If response is 1) y or Y, it places a Y in ANS and returns to calling procedure; 2) n or N, it places an N in ANS and returns to calling procedure; 3) h or H, it calls the HELP routine, places an H in ANS and returns to calling program; or 4) any other key—it displays PLEASE RESPOND YES OR NO and awaits a Y, N, H, y, n or h response. NOTE: Only the first character/line is processed. The rest is ignored.

```
1    6:D     1 (*$P*)PROCEDURE PREPKEY(HLP:INTEGER;MSG:STRING);
1    6:0     0   BEGIN
1    6:1     0     HELP:=HLP;
1    6:1     9     REPEAT
1    6:2     9       WRITE(MSG);
1    6:2    20       KEY;
1    6:1    22       UNTIL (ANS='Y') OR (ANS='N') OR (ORD(ANS)=27);
1    6:0    41     END;
1    6:0    56
```

PREPKEY displays a message then calls KEY to read a letter response from the keyboard. If a response is not Y, y, N, n, Yes or No, it redisplays the message and, once again, waits for a response.

```
201   1    7:D     1 (*$P*)PROCEDURE INLINE;
202   1    7:D     1   VAR
203   1    7:D     1     LONGLINE:STRING[125];
204   1    7:D    64     LINEOK:BOOLEAN;
205   1    7:D    65
206   1    7:0     0   BEGIN
207   1    7:1     0     REPEAT
208   1    7:2     0       READLN(LONGLINE);
209   1    7:2    19       LINEOK:=TRUE;
210   1    7:2    22       M:=LENGTH(LONGLINE);
211   1    7:2    29       IF M>80 THEN
212   1    7:3    36         BEGIN
213   1    7:4    36           WRITELN('**WARNING LINE CONTAINS OVER 80 CHARACTERS**');
214   1    7:4   100           WRITELN(' ');
215   1    7:4   118           PREPKEY(39,'DO YOU WISH TO TRUNCATE TO 80 CHARACTERS? ');
216   1    7:4   166           IF ANS='N' THEN
217   1    7:5   173             BEGIN
218   1    7:6   173               LINEOK:=FALSE;
219   1    7:6   176               WRITELN('PLEASE TYPE LINE AGAIN: ');
220   1    7:5   220               END
221   1    7:4   220             ELSE
222   1    7:5   222               M:=80;
223   1    7:3   226           END;
224   1    7:1   226       UNTIL LINEOK;
225   1    7:1   230     LINER:=COPY(LONGLINE,1,M);
226   1    7:0   248     END;
227   1    7:0   264
```

INLINE accepts up to 80 characters of text. If more than 80 characters are specified, it asks if it ought to ignore additional characters. If told to, it does. Otherwise, it allows analyst to re-enter the line.

```
 1   8:D    1 (*$P*)PROCEDURE BRANCHIN;
 1   8:0    0   BEGIN
 1   8:0    0     (*$I-*)
 1   8:1    0     RESET(PASSNODE,'PASSTHRU');
 1   8:1   18     I:=IORESULT;
 1   8:1   23     (*$I+*)
 1   8:1   23     IF I<>0 THEN
 1   8:2   30       BEGIN
 1   8:3   30         WRITELN('PASSTHRU FILE DOES NOT EXIST');
 1   8:3   78         WRITELN('  ******FATAL ERROR******');
 1   8:3  123         WRITELN('               ',I);
 1   8:3  167         ANYKEY;
 1   8:3  169         SETCHAIN('PGM1');
 1   8:3  179         EXIT(PROGRAM);
 1   8:2  183         END;
 1   8:1  183     GET(PASSNODE);
 1   8:1  190     CURSYS:=PASSNODE^.CURSYS;
 1   8:1  198     CURSP:=PASSNODE^.CURSP;
 1   8:1  206     CURSUB:=PASSNODE^.CURSUB;
 1   8:1  214     PAC:=PASSNODE^.PAC;
 1   8:1  220     NCURSYS:=PASSNODE^.NCURSYS;
 1   8:1  227     NCURSP:=PASSNODE^.NCURSP;
 1   8:1  234     NCURSUB:=PASSNODE^.NCURSUB;
 1   8:1  241     NPAC:=PASSNODE^.NPAC;
 1   8:1  248     CLOSE(PASSNODE);
 1   8:0  256     END;
 1   8:0  270
```

BRANCHIN gets information from the PASSTHRU file for use by this program.

```
1    9:D     1   (*$P*)PROCEDURE BRANCHOUT;
1    9:0     0     BEGIN
1    9:1     0       REWRITE(PASSNODE,'PASSTHRU');
1    9:1    20       PASSNODE^.FLAG1:=1;
1    9:1    26       PUT(PASSNODE);
1    9:1    33       CLOSE(PASSNODE,LOCK);
1    9:0    41       END;
1    9:0    54
1    9:0    54
1    9:0    54
1    9:0    54 (*$I #5:UTILITY.TEXT*)
1    9:0    54
```

BRANCHOUT loads the PASSTHRU file with appropriate data for use by called programs.

```
1    10:D     1 (*$P*)PROCEDURE OPENATTRIBUTESFILE;
1    10:0     0   BEGIN
1    10:1     0     NATTRLAST:=0;
1    10:1     4     (*$I-*)
1    10:1     4     RESET(ATTRIBUTES,NAMEATTRIBUTES);
1    10:1    15     (*$I+*)
1    10:1    15     I:=IORESULT;
1    10:1    20     IF I<>0 THEN
1    10:2    27       BEGIN
1    10:3    27         NATTRLAST:=-1;
1    10:3    32         WRITELN('There is no attributes file on disk');
1    10:2    87         END;
1    10:1    87     CLOSE(ATTRIBUTES);
1    10:0    96     END;
1    10:0   108
```

OPENATTRIBUTESFILE checks to see if there is an attributes file on the disk.

```
1    11:D     1 (*$P*)PROCEDURE OPENMEASURESFILE;
1    11:0     0   BEGIN
1    11:1     0     NMEASLAST:=0;
1    11:1     4     (*$I-*)
1    11:1     4     RESET(MEASURES,NAMEMEASURES);
1    11:1    15     (*$I+*)
1    11:1    15     I:=IORESULT;
1    11:1    20     IF I<>0 THEN
1    11:2    27       BEGIN
1    11:3    27         WRITELN('There is no measures file on disk');
1    11:3    80         NMEASLAST:=-1;
1    11:2    85         END;
1    11:1    85     CLOSE(MEASURES);
1    11:0    94     END;
1    11:0   106
```

OPENMEASURESFILE checks to see if there is a measures file on disk.

```
1   12:D      1 (*$P*)PROCEDURE READATTRFILE;
1   12:0      0   BEGIN
1   12:0      0   (*$I-*)
1   12:1      0   RESET(ATTRFILE,NAMEATCORE);
1   12:1     11   I:=IORESULT;
1   12:0     16   (*$I+*);
1   12:1     16   IF I<>0 THEN
1   12:2     23     BEGIN
1   12:3     23       REWRITE(ATTRFILE,NAMEATCORE);
1   12:3     36       FOR I:=1 TO NATTRIBUTES DO
1   12:4     52         BEGIN
1   12:5     52           ATTRCORE[I]:=0;
1   12:5     79           ATTRFILE^:=ATTRCORE[I];
1   12:5    107           PUT(ATTRFILE);
1   12:5    115           IF EOF(ATTRFILE) THEN
1   12:6    125             BEGIN
1   12:7    125               WRITELN('OUT OF DISK SPACE');
1   12:7    162               ANYKEY;
1   12:6    164               END;
1   12:4    164           END;
1   12:3    174         ATTRLAST:=0;
1   12:3    189         NATTRLAST:=0;
1   12:3    193         ATTRFILE^:=ATTRLAST;
1   12:3    209         PUT(ATTRFILE);
1   12:3    217         CLOSE(ATTRFILE,LOCK);
1   12:2    226         END
1   12:1    226     ELSE
1   12:2    228       BEGIN
1   12:3    228         FOR I:=1 TO NATTRIBUTES DO
1   12:4    244           BEGIN
1   12:5    244             GET(ATTRFILE);
1   12:5    252             ATTRCORE[I]:=ATTRFILE^;
1   12:4    280             END;
1   12:3    290         GET(ATTRFILE);
1   12:3    298         ATTRLAST:=ATTRFILE^;
1   12:3    314         NATTRLAST:=TRUNC(ATTRLAST);
1   12:3    327         CLOSE(ATTRFILE);
1   12:2    336         END;
1   12:0    336     END;
1   12:0    354
```

READATTRFILE reads the index to the attributes file from the disk file ATTRFILE and places it into the array ATTRCORE.

```
1   13:D    1 (*$P*)PROCEDURE READMEASFILE;
1   13:0    0   BEGIN
1   13:0    0   (*$I-*)
1   13:1    0   RESET(MEASFILE,NAMEMECORE);
1   13:1   11   I:=IORESULT;
1   13:0   16   (*$I+*);
1   13:1   16   IF I<>0 THEN
1   13:2   23     BEGIN
1   13:3   23       REWRITE(MEASFILE,NAMEMECORE);
1   13:3   36       FOR I:=1 TO NMEASURES DO
1   13:4   52         BEGIN
1   13:5   52           MEASCORE[I]:=0;
1   13:5   79           MEASFILE^:=MEASCORE[I];
1   13:5  107           PUT(MEASFILE);
1   13:5  115           IF EOF(MEASFILE) THEN
1   13:6  125             BEGIN
1   13:7  125               WRITELN('OUT OF DISK SPACE');
1   13:7  162               ANYKEY;
1   13:6  164               END;
1   13:4  164           END;
1   13:3  174         MEASLAST:=0;
1   13:3  189         NMEASLAST:=0;
1   13:3  193         MEASFILE^:=MEASLAST;
1   13:3  209         PUT(MEASFILE);
1   13:3  217         CLOSE(MEASFILE,LOCK);
1   13:2  226         END
1   13:1  226     ELSE
1   13:2  228       BEGIN
1   13:3  228         FOR I:=1 TO NMEASURES DO
1   13:4  244           BEGIN
1   13:5  244             GET(MEASFILE);
1   13:5  252             MEASCORE[I]:=MEASFILE^;
1   13:4  280             END;
1   13:3  290         GET(MEASFILE);
1   13:3  298         MEASLAST:=MEASFILE^;
1   13:3  314         NMEASLAST:=TRUNC(MEASLAST);
1   13:3  327         CLOSE(MEASFILE);
1   13:2  336         END;
1   13:0  336     END;
1   13:0  354
```

READMEASFILE reads the index to the measures file from the disk file MEASFILE and places it into the array MEASCORE.

```
1    14:D     1 (*$P*)PROCEDURE CLOSECOREFILE;
1    14:0     0   BEGIN
1    14:1     0     RESET(COREFILE,CORENAME);
1    14:1    13     FOR I:=1 TO 300 DO
1    14:2    29       BEGIN
1    14:3    29         COREFILE^:=CORE[I];
1    14:3    57         PUT(COREFILE);
1    14:2    65         END;
1    14:1    75     CORELAST:=NCORELAST;
1    14:1    92     COREFILE^:=CORELAST;
1    14:1   108     PUT(COREFILE);
1    14:1   116     CLOSE(COREFILE);
1    14:0   125     END;
1    14:0   140
```

CLOSECOREFILE copies the index to the performance items from the array core to the disk file COREFILE.

```
 1   15:D     1 (*$P*)PROCEDURE CLOSEATTRFILE;
 1   15:0     0   BEGIN
 1   15:1     0     RESET(ATTRFILE,NAMEATCORE);
 1   15:1    13     FOR I:=1 TO NATTRIBUTES DO
 1   15:2    29       BEGIN
 1   15:3    29         ATTRFILE^:=ATTRCORE[I];
 1   15:3    57         PUT(ATTRFILE);
 1   15:2    65         END;
 1   15:1    75     ATTRLAST:=NATTRLAST;
 1   15:1    92     ATTRFILE^:=ATTRLAST;
 1   15:1   108     PUT(ATTRFILE);
 1   15:1   116     CLOSE(ATTRFILE);
 1   15:0   125     END;
 1   15:0   140
```

CLOSEATTRFILE copies the index to the attribute file from the array ATTRCORE to the disk file ATTRFILE.

```
1   16:D     1 (*$P*)PROCEDURE CLOSEMEASFILE;
1   16:0     0   BEGIN
1   16:1     0     RESET(MEASFILE,NAMEMECORE);
1   16:1    13     FOR I:=1 TO NMEASURES DO
1   16:2    29       BEGIN
1   16:3    29         MEASFILE^:=MEASCORE[I];
1   16:3    57         PUT(MEASFILE);
1   16:2    65         END;
1   16:1    75     MEASLAST:=NMEASLAST;
1   16:1    92     MEASFILE^:=MEASLAST;
1   16:1   108     PUT(MEASFILE);
1   16:1   116     CLOSE(MEASFILE);
1   16:0   125     END;
1   16:0   140
```

CLOSEMEASFILE copies the index to the measures file from the array MEASCORE to the disk file MEASFILE.

```
1    17:D      1 (*$P*) PROCEDURE OPENDATAFILE;
1    17:0      0   BEGIN
1    17:1      0     NCORELAST:=0;
1    17:1      4     (*$I-*)
1    17:1      4     RESET(DATANODE,DATANAME);
1    17:1     15     (*$I+*)
1    17:1     15     I:=IORESULT;
1    17:1     20     IF I<>0 THEN
1    17:2     27       BEGIN
1    17:3     27         WRITELN('There is no performance items file on disk');
1    17:3     89         NCORELAST:=-1;
1    17:2     94         END;
1    17:1     94     CLOSE(DATANODE);
1    17:0    103     END;
1    17:0    116
```

OPENDATAFILE determines whether there are any performance items on the disk.

```
434   1   18:D     1 (*$P*)PROCEDURE DEFINEASPECTS;
435   1   18:0     0   BEGIN
436   1   18:1     0     ASPECT[1]:='Potentialities';
437   1   18:1    30     ASPECT[2]:='Processes';
438   1   18:1    55     ASPECT[3]:='Products';
439   1   18:1    79     ASPECT[4]:='Environment';
440   1   18:1   106     ASPECT[5]:='Constraints';
441   1   18:1   133     NASPECT[1]:=1;
442   1   18:1   146     NASPECT[2]:=2;
443   1   18:1   159     NASPECT[3]:=3;
444   1   18:1   172     NASPECT[4]:=4;
445   1   18:1   185     NASPECT[5]:=5;
446   1   18:0   198     END;
447   1   18:0   210
```

DEFINEASPECTS tells computer the name assigned to each of the aspects.

```
1   19:D     1 (*$P*)PROCEDURE READCOREFILE;
1   19:0     0   BEGIN
1   19:0     0   (*$I-*)
1   19:1     0   RESET(COREFILE,CORENAME);
1   19:1    11   I:=IORESULT;
1   19:1    16   (*$I+*)
1   19:1    16   IF I<>0 THEN
1   19:2    23     BEGIN
1   19:3    23       IF I=9 THEN
1   19:4    30         BEGIN
1   19:5    30           PAGE(OUTPUT);
1   19:5    40           WRITELN('THE APMDISK IS NOT MOUNTED');
1   19:5    86           WRITELN('');
1   19:5   106           WRITELN('PLEASE PLACE IT IN DRIVE #2');
1   19:5   153           ANYKEY;
1   19:5   155           READCOREFILE;
1   19:5   157           EXIT(READCOREFILE)
1   19:4   161           END
1   19:3   161         ELSE
1   19:4   163           BEGIN
1   19:5   163             WRITELN('COREFILE DOES NOT EXIST');
1   19:5   206             WRITELN('  ****FATAL ERROR****  ');
1   19:5   249             WRITELN('              ',I);
1   19:5   292             ANYKEY;
1   19:5   294             BRANCHOUT;
1   19:5   296             SETCHAIN('GREETING');
1   19:5   310             EXIT(PROGRAM);
1   19:4   314             END;
1   19:2   314           END
1   19:1   314       ELSE
1   19:2   316         FOR I:=1 TO 300 DO
1   19:3   332           BEGIN
1   19:4   332             GET(COREFILE);
1   19:4   340             CORE[I]:=COREFILE^;
1   19:3   348             END;
1   19:1   378         GET(COREFILE);
1   19:1   386         CORELAST:=COREFILE^;
1   19:1   402         NCORELAST:=TRUNC(CORELAST);
1   19:1   415         CLOSE(COREFILE)
1   19:0   424       END;
1   19:0   444
```

READCOREFILE copies performance items from the disk file DATANODE to the CORE file.

```
1   20:D     1 (*$P*)PROCEDURE SORTATTRFILE;
1   20:0     0   BEGIN
1   20:1     0     IF NATTRLAST<2 THEN
1   20:2     7       EXIT(SORTATTRFILE);
1   20:1    11     FOR I:=1 TO NATTRIBUTES DO
1   20:2    27       ATTR2[I]:=I;
1   20:1    56     I:=2;
1   20:1    60     REPEAT
1   20:2    60       IF ATTRCORE[I]<ATTRCORE[I-1] THEN
1   20:3   105         BEGIN
1   20:4   105           TEMPX:=ATTRCORE[I];
1   20:4   133           ATTRCORE[I]:=ATTRCORE[I-1];
1   20:4   175           ATTRCORE[I-1]:=TEMPX;
1   20:4   205           TEMP2:=ATTR2[I];
1   20:4   224           ATTR2[I]:=ATTR2[I-1];
1   20:4   258           ATTR2[I-1]:=TEMP2;
1   20:4   279           IF I>2 THEN
1   20:5   286             I:=I-1;
1   20:3   294           END
1   20:2   294         ELSE
1   20:3   296           I:=I+1;
1   20:1   304       UNTIL I>NATTRLAST;
1   20:0   313     END;
1   20:0   332
```

SORTATTRFILE forms an array ATTR2 which contains a sorted permutation vecyor referencing the attributes file [sorted by numerical value of the index]--sort attributes index into assending numerical order.

```
1    21:D     1 (*$P*)PROCEDURE SORTMEASFILE;
1    21:0     0   BEGIN
1    21:1     0     IF NMEASLAST<2 THEN
1    21:2     7       EXIT(SORTMEASFILE);
1    21:1    11     FOR I:=1 TO NMEASURES DO
1    21:2    27       MEAS2[I]:=I;
1    21:1    56     I:=2;
1    21:1    60     REPEAT
1    21:2    60       IF MEASCORE[I]<MEASCORE[I-1] THEN
1    21:3   105         BEGIN
1    21:4   105           TEMPX:=MEASCORE[I];
1    21:4   133           MEASCORE[I]:=MEASCORE[I-1];
1    21:4   175           MEASCORE[I-1]:=TEMPX;
1    21:4   205           TEMP2:=MEAS2[I];
1    21:4   224           MEAS2[I]:=MEAS2[I-1];
1    21:4   258           MEAS2[I-1]:=TEMP2;
1    21:4   279           IF I>2 THEN
1    21:5   286             I:=I-1;
1    21:3   294           END
1    21:2   294         ELSE
1    21:3   296           I:=I+1;
1    21:1   304       UNTIL I>NMEASLAST;
1    21:0   313     END;
1    21:0   332
```

SORTMEASFILE forms an array MEAS2 which contains a sorted permutation vector referencing the measures file—sort measures index into ascending numerical order.

```
537   1   22:D     1 (*$P*)PROCEDURE SORTCOREFILE;
538   1   22:0     0   BEGIN
539   1   22:1     0     IF NCORELAST<2 THEN
540   1   22:2     7       EXIT(SORTCOREFILE);
541   1   22:1    11     FOR I:=1 TO 300 DO
542   1   22:2    27       CORE2[I]:=I;
543   1   22:1    56     I:=2;
544   1   22:1    60     REPEAT
545   1   22:2    60       IF CORE[I]<CORE[I-1] THEN
546   1   22:3   105         BEGIN
547   1   22:4   105           TEMP:=CORE[I];
548   1   22:4   133           CORE[I]:=CORE[I-1];
549   1   22:4   175           CORE[I-1]:=TEMP;
550   1   22:4   205           TEMP2:=CORE2[I];
551   1   22:4   224           CORE2[I]:=CORE2[I-1];
552   1   22:4   258           CORE2[I-1]:=TEMP2;
553   1   22:4   279           IF I>2 THEN
554   1   22:5   286             I:=I-1;
555   1   22:3   294           END
556   1   22:2   294         ELSE
557   1   22:3   296           I:=I+1;
558   1   22:1   304       UNTIL I>NCORELAST;
559   1   22:0   313     END;
560   1   22:0   332
```

SORTCOREFILE forms an array CORE2 which contains a sorted permutation vector referencing the core file.

```
1   23:D     1 (*$P*)PROCEDURE COPYATTRIBUTES;
1   23:0     0   BEGIN
1   23:1     0     RESET(ATTRIBUTES,NAMEATTRIBUTES);
1   23:1    13     REWRITE(TEMPATTRIBUTES,NAMETEMPORARY);
1   23:1    26     IF IORESULT<>0 THEN
1   23:2    32       BEGIN
1   23:3    32         WRITELN('PROBLEM CREATING TEMPORARY DATASET');
1   23:3    86         ANYKEY;
1   23:2    88         END;
1   23:1    88     FOR I:=1 TO NATTRLAST DO
1   23:2   104       BEGIN
1   23:3   104         SEEK(ATTRIBUTES,ATTR2[I]);
1   23:3   128         GET(ATTRIBUTES);
1   23:3   136         SEEK(TEMPATTRIBUTES,I);
1   23:3   147         TEMPATTRIBUTES^:=ATTRIBUTES^;
1   23:3   155         PUT(TEMPATTRIBUTES);
1   23:2   163         END;
1   23:1   173     FOR I:=1 TO NATTRLAST DO
1   23:2   189       BEGIN
1   23:3   189         SEEK(TEMPATTRIBUTES,I);
1   23:3   200         GET(TEMPATTRIBUTES);
1   23:3   208         SEEK(ATTRIBUTES,I);
1   23:3   219         ATTRIBUTES^:=TEMPATTRIBUTES^;
1   23:3   227         PUT(ATTRIBUTES);
1   23:2   235         END;
1   23:1   245     CLOSE(ATTRIBUTES);
1   23:1   254     CLOSE(TEMPATTRIBUTES);
1   23:0   263     END;
1   23:0   280
```

COPYATTRIBUTES copies the attributes file from the disk file to the temporary storage disk [sorting attributes into numerical order] and temporary disk back to the usual storage disk.

```
1   24:D     1 (*$P*)PROCEDURE COPYMEASURES;
1   24:0     0   BEGIN
1   24:1     0     RESET(MEASURES,NAMEMEASURES);
1   24:1    13     REWRITE(TEMPMEASURES,NAMETEMPORARY);
1   24:1    26     IF IORESULT<>0 THEN
1   24:2    32       BEGIN
1   24:3    32         WRITELN('PROBLEM CREATING TEMPORARY DATASET');
1   24:3    86         ANYKEY;
1   24:2    88         END;
1   24:1    88     FOR I:=1 TO NMEASLAST DO
1   24:2   104       BEGIN
1   24:3   104         SEEK(MEASURES,MEAS2[I]);
1   24:3   128         GET(MEASURES);
1   24:3   136         SEEK(TEMPMEASURES,I);
1   24:3   147         TEMPMEASURES^:=MEASURES^;
1   24:3   155         PUT(TEMPMEASURES);
1   24:2   163         END;
1   24:1   173     FOR I:=1 TO NMEASLAST DO
1   24:2   189       BEGIN
1   24:3   189         SEEK(TEMPMEASURES,I);
1   24:3   200         GET(TEMPMEASURES);
1   24:3   208         SEEK(MEASURES,I);
1   24:3   219         MEASURES^:=TEMPMEASURES^;
1   24:3   227         PUT(MEASURES);
1   24:2   235         END;
1   24:1   245     CLOSE(MEASURES);
1   24:1   254     CLOSE(TEMPMEASURES);
1   24:0   263     END;
1   24:0   280
```

COPYMEASURES copies the measures file from the disk file to the temporary storage disk [sorting measures into numeric order] and copying the items from the temporary disk back to the usual storage disk.

```
1   25:D     1 (*$P*)PROCEDURE COPYCORE;
1   25:0     0   BEGIN
1   25:1     0     RESET(DATANODE,DATANAME);
1   25:1    13     REWRITE(TEMPDATA,NAMETEMPORARY);
1   25:1    26     IF IORESULT<>0 THEN
1   25:2    32       BEGIN
1   25:3    32         WRITELN('PROBLEM CREATING TEMPORARY DATASET');
1   25:3    86         ANYKEY;
1   25:2    88         END;
1   25:1    88     FOR I:=1 TO NCORELAST DO
1   25:2   104       BEGIN
1   25:3   104         SEEK(DATANODE,CORE2[I]);
1   25:3   128         GET(DATANODE);
1   25:3   136         SEEK(TEMPDATA,I);
1   25:3   147         TEMPDATA^:=DATANODE^;
1   25:3   155         PUT(TEMPDATA);
1   25:2   163         END;
1   25:1   173     FOR I:=1 TO NCORELAST DO
1   25:2   189       BEGIN
1   25:3   189         SEEK(TEMPDATA,I);
1   25:3   200         GET(TEMPDATA);
1   25:3   208         SEEK(DATANODE,I);
1   25:3   219         DATANODE^:=TEMPDATA^;
1   25:3   227         PUT(DATANODE);
1   25:2   235         END;
1   25:1   245     CLOSE(DATANODE);
1   25:1   254     CLOSE(TEMPDATA);
1   25:0   263     END;
1   25:0   280
```

COPYCORE copies the core file from the disk file to the temporary storage disk [sorting items into numeric order] and copying the items from the temporary disk back to the usual storage disk.

```
1   26:D     1 (*$P*)PROCEDURE REMOVEATTRIBUTES;
1   26:0     0   BEGIN
1   26:1     0     SEEK(ATTRIBUTES,J);
1   26:1    11     FOR L:=1 TO 6 DO
1   26:2    25       ATTRIBUTES^.NDESCRIPTOR[L]:=0;
1   26:1    50     ATTRIBUTES^.DESCRIPTOR:='';
1   26:1    60     PUT(ATTRIBUTES);
1   26:1    68     ATTRCORE[J]:=0;
1   26:0    95     END;
1   26:0   110
```

REMOVEATTRIBUTES removes an attribute from attributes file.

```
 1   27:D     1 (*$P*)PROCEDURE REMOVEMEASURES;
 1   27:0     0   BEGIN
 1   27:1     0     SEEK(MEASURES,J);
 1   27:1    11     FOR L:=1 TO 6 DO
 1   27:2    25       MEASURES^.NDESCRIPTOR[L]:=0;
 1   27:1    50     MEASURES^.DESCRIPTOR:='';
 1   27:1    60     PUT(MEASURES);
 1   27:1    68     MEASCORE[J]:=0;
 1   27:0    95     END;
 1   27:0   110
```

REMOVEMEASURES removes a measure from measures file.

```
  1   28:D     1 (*$P*)PROCEDURE REMOVEDATA;
  1   28:0     0   BEGIN
  1   28:1     0     SEEK(DATANODE,J);
  1   28:1    11     FOR L:=1 TO 4 DO
  1   28:2    25       DATANODE^.NTAXA[L]:=0;
  1   28:1    50     DATANODE^.TAXA:='';
  1   28:1    60     PUT(DATANODE);
  1   28:1    68     CORE[J]:=0;
  1   28:0    95     END;
  1   28:0   110
```

REMOVEDATA removes a performance item from data file.

```
 1   29:D     1 (*$P*)PROCEDURE COMPACTATTRIBUTES;
 1   29:0     0   BEGIN
 1   29:1     0     RESET(ATTRIBUTES,NAMEATTRIBUTES);
 1   29:1    13     M:=0;
 1   29:1    17     I:=0;
 1   29:1    21     REPEAT
 1   29:2    21       I:=I+1;
 1   29:2    29       REPEAT
 1   29:3    29         IF ATTRCORE[I+M]=0 THEN
 1   29:4    63           M:=M+1;
 1   29:3    71           J:=I+M;
 1   29:3    81           IF J>NATTRLAST THEN
 1   29:4    90             BEGIN
 1   29:5    90               I:=I+1;
 1   29:5    98               FOR J:=I TO NATTRLAST DO
 1   29:6   116                 REMOVEATTRIBUTES;
 1   29:5   128               NATTRLAST:=NATTRLAST-M;
 1   29:5   138               CLOSE(ATTRIBUTES);
 1   29:5   147               EXIT(COMPACTATTRIBUTES);
 1   29:4   151               END;
 1   29:2   151         UNTIL ATTRCORE[I+M]<>0;
 1   29:2   185       ATTRCORE[I]:=ATTRCORE[I+M];
 1   29:2   229       J:=I+M;
 1   29:2   239       SEEK(ATTRIBUTES,J);
 1   29:2   250       GET(ATTRIBUTES);
 1   29:2   258       SEEK(ATTRIBUTES,I);
 1   29:2   269       PUT(ATTRIBUTES);
 1   29:1   277       UNTIL J=NATTRLAST;
 1   29:1   286     I:=I+1;
 1   29:1   294     FOR J:=I TO NATTRLAST DO
 1   29:2   312       REMOVEATTRIBUTES;
 1   29:1   324     NATTRLAST:=NATTRLAST-M;
 1   29:1   334     CLOSE(ATTRIBUTES);
 1   29:0   343     END;
 1   29:0   364
```

COMPACTATTRIBUTES packs attribute data set so that all blank entries are pushed to the end of the data set.

```
 1   30:D     1 (*$P*)PROCEDURE COMPACTMEASURES;
 1   30:0     0   BEGIN
 1   30:1     0     RESET(MEASURES,NAMEMEASURES);
 1   30:1    13     M:=0;
 1   30:1    17     I:=0;
 1   30:1    21     REPEAT
 1   30:2    21       I:=I+1;
 1   30:2    29       REPEAT
 1   30:3    29         IF MEASCORE[I+M]=0 THEN
 1   30:4    63           M:=M+1;
 1   30:3    71           J:=I+M;
 1   30:3    81           IF J>NMEASLAST THEN
 1   30:4    90             BEGIN
 1   30:5    90               I:=I+1;
 1   30:5    98               FOR J:=I TO NMEASLAST DO
 1   30:6   116                 REMOVEMEASURES;
 1   30:5   128               NMEASLAST:=NMEASLAST-M;
 1   30:5   138               CLOSE(MEASURES);
 1   30:5   147               EXIT(COMPACTMEASURES);
 1   30:4   151               END;
 1   30:2   151         UNTIL MEASCORE[I+M]<>0;
 1   30:2   185       MEASCORE[I]:=MEASCORE[I+M];
 1   30:2   229       J:=I+M;
 1   30:2   239       SEEK(MEASURES,J);
 1   30:2   250       GET(MEASURES);
 1   30:2   258       SEEK(MEASURES,I);
 1   30:2   269       PUT(MEASURES);
 1   30:1   277       UNTIL J=NMEASLAST;
 1   30:1   286     I:=I+1;
 1   30:1   294     FOR J:=I TO NMEASLAST DO
 1   30:2   312       REMOVEMEASURES;
 1   30:1   324     NMEASLAST:=NMEASLAST-M;
 1   30:1   334     CLOSE(MEASURES);
 1   30:0   343     END;
 1   30:0   364
```

COMPACTMEASURES packs measures data set more efficiently so that all blank entries are pushed to the end of the data set.

```
  1    31:D      1 (*$P*)PROCEDURE COMPACTCORE;
  1    31:0      0   BEGIN
  1    31:1      0     RESET(DATANODE,DATANAME);
  1    31:1     13     M:=0;
  1    31:1     17     I:=0;
  1    31:1     21     REPEAT
  1    31:2     21       I:=I+1;
  1    31:2     29       REPEAT
  1    31:3     29         IF CORE[I+M]=0 THEN
  1    31:4     63           M:=M+1;
  1    31:3     71           J:=I+M;
  1    31:3     81           IF J>NCORELAST THEN
  1    31:4     90             BEGIN
  1    31:5     90               I:=I+1;
  1    31:5     98               FOR J:=I TO NCORELAST DO
  1    31:6    116                 REMOVEDATA;
  1    31:5    128               NCORELAST:=NCORELAST-M;
  1    31:5    138               CLOSE(DATANODE);
  1    31:5    147               EXIT(COMPACTCORE);
  1    31:4    151               END;
  1    31:2    151         UNTIL CORE[I+M]<>0;
  1    31:2    185       CORE[I]:=CORE[I+M];
  1    31:2    229       J:=I+M;
  1    31:2    239       SEEK(DATANODE,J);
  1    31:2    250       GET(DATANODE);
  1    31:2    258       SEEK(DATANODE,I);
  1    31:2    269       PUT(DATANODE);
  1    31:1    277       UNTIL J=NCORELAST;
  1    31:1    286     I:=I+1;
  1    31:1    294     FOR J:=I TO NCORELAST DO
  1    31:2    312       REMOVEDATA;
  1    31:1    324     NCORELAST:=NCORELAST-M;
  1    31:1    334     CLOSE(DATANODE);
  1    31:0    343     END;
  1    31:0    364
```

COMPACTCORE packs performance item data set more efficiently so that all blank entries are pushed to the end of the data set.

```
783 1   32:D     1   (*$P*)PROCEDURE DONOT;
784 1   32:0     0     BEGIN
785 1   32:1     0       WRITELN('  This file contains less than 2 items, thus it will not be packed');
786 1   32:0    84       END;
787 1   32:0    98
```

DONOT displays warning message that file will not be packed.

```
 1   33:D     1   (*$P*)PROCEDURE PROPERUTLDISK;
 1   33:0     0     BEGIN
 1   33:1     0       REPEAT
 1   33:1     0         (*$I-*)
 1   33:2     0         RESET(TEMPDATA,'APMUTL:TEMPORARY');
 1   33:2    27         (*$I+*)
 1   33:2    27         K:=IORESULT;
 1   33:2    32         IF K=0 THEN
 1   33:3    39           CLOSE(TEMPDATA);
 1   33:2    48         IF K=9 THEN
 1   33:3    55           BEGIN
 1   33:4    55             WRITELN('Please place the APM UTILITY disk in drive # 1');
 1   33:4   121             ANYKEY;
 1   33:3   123             END;
 1   33:1   123         UNTIL K<>9;
 1   33:0   130       END;
 1   33:0   144
```

PROPERUTILDISK checks to be sure APMUTL (the disk used for temporary storage) is in Drive #1.

```
1   34:D     1   (*$P*)PROCEDURE PROPERMAINDISK;
1   34:0     0     BEGIN
1   34:1     0       REPEAT
1   34:1     0         (*$I-*)
1   34:2     0         RESET(TEMPDATA,'APMSYS:TEMPORARY');
1   34:2    27         (*$I+*)
1   34:2    27         K:=IORESULT;
1   34:2    32         IF K=0 THEN
1   34:3    39           CLOSE(TEMPDATA);
1   34:2    48         IF K=9 THEN
1   34:3    55           BEGIN
1   34:4    55             WRITELN('Please place the APM SYSTEM disk in drive # 1');
1   34:4   120             ANYKEY;
1   34:3   122             END;
1   34:1   122         UNTIL K<>9;
1   34:0   129       END;
1   34:0   144
```

PROPERMAINDISK checks to be sure the APYSYS disk has been returned to Drive #1 before returning to select a different analytic procedure.

```
822 1 35:D   1 (*$P*)PROCEDURE ASSIGNNAMES;
823 1 35:0   0   BEGIN
824 1 35:1   0     APMDSK:=CONCAT(COPY(CURSYS,1,2),COPY(CURSP,1,2),COPY(CURSUB,1,2),':');
825 1 35:1  86     NAMETEMPORARY:=CONCAT('APMUTL:TEMPORARY');
826 1 35:1 121     NAMEATCORE:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'AC');
827 1 35:1 217     NAMEATTRIBUTES:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),
                   'AT');
828 1 35:1 313     NAMEMECORE:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'MC');
829 1 35:1 409     NAMEMEASURES:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),
                   'ME');
830 1 35:1 505     CORENAME:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'CO');
831 1 35:1 601     DATANAME:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'FI');
832 1 35:1 697     ISSUENAME:=CONCAT(APMDSK,(COPY(CURSYS,1,4)),COPY(CURSP,1,4),(COPY(CURSUB,1,4)),'IS');
833 1 35:0 793     END;
834 1 35:0 806
```

ASSIGNNAMES determines file names based upon system class, system and subsystem names.

```
1   1:0     0   (*$P*)BEGIN
1   1:0     0     (*$N+*)
1   1:1     0     PAGE(OUTPUT);
1   1:1   141     BRANCHIN;
1   1:1   143     PROPERUTLDISK;
1   1:1   145     WRITELN('I am going to sort and pack all data sets, but I am slow,',chr(13),
1   1:1   224            '    so please take a coffee break at this time',chr(13),chr(13));
1   1:1   309     NISSUES:=5;
1   1:1   313     NMEASURES:=400;
1   1:1   319     NATTRIBUTES:=200;
1   1:1   325
1   1:1   325     ASSIGNNAMES;
1   1:1   327
1   1:1   327     WRITELN('Processing attributes');
1   1:1   368     OPENATTRIBUTESFILE;
1   1:1   370     IF NATTRLAST<>-1 THEN
1   1:2   378       BEGIN
1   1:3   378         READATTRFILE;
1   1:3   380         IF NATTRLAST>2 THEN
1   1:4   387           BEGIN
1   1:5   387             COMPACTATTRIBUTES;
1   1:5   389             SORTATTRFILE;
1   1:5   391             COPYATTRIBUTES;
1   1:5   393             CLOSEATTRFILE;
1   1:4   395             END
1   1:3   395           ELSE
1   1:4   397             DONOT;
1   1:2   399         END;
1   1:2   399
1   1:1   399   WRITELN('Processing measures');
1   1:1   438   OPENMEASURESFILE;
1   1:1   440   IF NMEASLAST<>-1 THEN
1   1:2   448     BEGIN
1   1:3   448       READMEASFILE;
1   1:3   450       IF NMEASLAST>2 THEN
1   1:4   457         BEGIN
1   1:5   457           COMPACTMEASURES;
1   1:5   459           SORTMEASFILE;
1   1:5   461           COPYMEASURES;
1   1:5   463           CLOSEMEASFILE;
```

Main Program: Governs overall packing and sorting of attributes, measures and performance items.

```
1   1:4   465           END
1   1:3   465         ELSE
1   1:4   467           DONOT;
1   1:2   469       END;
1   1:2   469
1   1:1   469   WRITELN('Processing performance items');
1   1:1   517   OPENDATAFILE;
1   1:1   519   IF NCORELAST<>-1 THEN
1   1:2   527     BEGIN
1   1:3   527       READCOREFILE;
1   1:3   529       IF NCORELAST>2 THEN
1   1:4   536         BEGIN
1   1:5   536           COMPACTCORE;
1   1:5   538           SORTCOREFILE;
1   1:5   540           COPYCORE;
1   1:5   542           CLOSECOREFILE;
1   1:4   544           END
1   1:3   544         ELSE
1   1:4   546           DONOT;
1   1:2   548       END;
1   1:2   548
1   1:1   548     WRITELN(chr(13),'Data sets are packed and sorted',chr(13),chr(13),
1   1:1   621             '....so long for now');
1   1:1   660
1   1:1   660     PROPERMAINDISK;
1   1:1   662
1   1:1   662     BRANCHOUT;
1   1:1   664
1   1:1   664     SETCHAIN('GREETING');
1   1:1   678
1   1:0   678     END.
```

See previous page for program description.

MISCELLANEOUS

Two of the programs in this system are present on the BOOT disk. They ask the analyst to set up the APM System disk. The remaining programs in this section are not actually part of the APM Demonstration Package. They are used to prepare data sets for use with the APM System.

```
 1    1:D     1 (*$L PRINTER:*)
 1    1:D     1 (*SYSTEM.STARTUP TELLS THE ANALYST TO PLACE THE REAL APM SYSTEM DISK IN DRIVE # 1*)
 1    1:D     1 (*RONALD G. SHAPIRO                    V 2.0                    10/25/82*)
 1    1:D     1
 1    1:D     1 PROGRAM STARTUP;
28    1:D     3
28    2:D     1   PROCEDURE SETCHAIN(TYTLE:STRING);
28    3:D     1   PROCEDURE SETCVAL(VAL:STRING);
28    4:D     1   PROCEDURE GETCVAL(VAR VAL:STRING);
28    5:D     1   PROCEDURE SWAPON;
28    6:D     1   PROCEDURE SWAPOFF;
28    6:D     1
 1    1:D     1 USES CHAINSTUFF;
 1    1:D     3 VAR
 1    1:D     3   X:CHAR;
 1    1:0     0 BEGIN
 1    1:1     0   PAGE(OUTPUT);
 1    1:1    15   WRITELN('Please insert the APM SYSTEM DISK in drive # 1');
 1    1:1    81   WRITELN('  Then press any Key to continue');
 1    1:1   133   (*$I-*)
 1    1:1   133   READ(X);
 1    1:1   141   (*$I+*)
 1    1:1   141   SETCHAIN('GREETING');
 1    1:1   155   EXIT(PROGRAM);
 1    1:0   159 END.
```

SYSTEMPOINTSARTUP is present on the Boot disk. It simply tells the analyst when it is time to set up the APM System disk and press a key to continue. When the disk is set up, it transfers control to the GREETING program.

```
 1    1:D     1 (*$L PRINTER:*)
 1    1:D     1 (*GREETSHORT TELLS THE ANALYST TO PLACE THE REAL APM SYSTEM DISK IN DRIVE#1*)
 1    1:D     1 (*RONALD G. SHAPIRO              V 2.0                          10/19/82*)
 1    1:D     1
 1    1:D     1 PROGRAM GREETING;
28    1:D     3
28    2:D     1   PROCEDURE SETCHAIN(TYTLE:STRING);
28    3:D     1   PROCEDURE SETCVAL(VAL:STRING);
28    4:D     1   PROCEDURE GETCVAL(VAR VAL:STRING);
28    5:D     1   PROCEDURE SWAPON;
28    6:D     1   PROCEDURE SWAPOFF;
28    6:D     1
 1    1:D     1 USES CHAINSTUFF;
 1    1:D     3 VAR
 1    1:D     3   X:CHAR;
 1    1:0     0 BEGIN
 1    1:1     0   PAGE(OUTPUT);
 1    1:1    15   WRITELN('Please insert the APM SYSTEM DISK in drive # 1');
 1    1:1    81   WRITELN('  Then press any key to continue');
 1    1:1   133   (*$I-*)
 1    1:1   133   READ(X);
 1    1:1   141   (*$I+*)
 1    1:1   141   SETCHAIN('GREETING');
 1    1:1   155   EXIT(PROGRAM);
 1    1:0   159 END.
```

GREETSHORT—If the analyst fails to set up the system disk, then the GREETSHORT program is executed. It, once again, asks the analyst to set up the system disk and press a key. The only ways to exit from this program are to set up the system disk, press Control Reset or turn the computer off.

```
1   1:D     1 (*$L PRINTER: *)
1   1:D     1 PROGRAM BLOCKINSTRUCTIONS;
1   1:D     3 (*Program to take text instructions file and convert it to blocked instr file*)
1   1:D     3 (*  After editing file, X BLOCKINSTR.  At the pause, place this disk in*)
1   1:D     3 (*  Drive #1 and place the APM UTIL disk in Drive #2.  Press any key.*)
1   1:D     3 (*  Within a few minutes, files will be blocked.*)
1   1:D     3 (*Note:
1   1:D     3     Each frame of text must be exactly 20 lines long in the text file!*)
1   1:D     3 (*Ronald G. Shapiro               V2.0                     10/25/82*)
1   1:D     3
1   1:D     3 TYPE
1   1:D     3 INSTRFILE=RECORD
1   1:D     3 LINE:ARRAY[1..20] OF STRING[80];
1   1:D     3 END;
1   1:D     3
1   1:D     3 VAR
1   1:D     3 INSTFILE:FILE OF INSTRFILE;
1   1:D  1123 ORIGINST:TEXT;
1   1:D  1424 I,J,K,L,M,N:INTEGER;
1   1:D  1430 LINE:STRING[80];
1   1:D  1471 A:CHAR;
1   1:D  1472
1   1:0     0 BEGIN
1   1:1     0 writeln('press any key to begin');
1   1:1    68 read(a);
1   1:1    79 REWRITE(INSTFILE,'apmutl:INSTRUCT');
1   1:1   106 SEEK(INSTFILE,1);
1   1:1   114 PUT(INSTFILE);
1   1:1   121 CLOSE(INSTFILE,LOCK);
1   1:1   129 RESET(INSTFILE,'apmutl:INSTRUCT');
1   1:1   156 RESET(ORIGINST,'apmpg4:INSTR.TEXT');
1   1:1   186
1   1:1   186 J:=1;
1   1:1   190 REPEAT
1   1:2   190   J:=J+1;
1   1:2   198   FOR I:=1 TO 20 DO
1   1:3   215     BEGIN
1   1:3   215       (*$R-*)
1   1:3   215       (*$I-*)
1   1:4   215       READLN(ORIGINST,LINE);
1   1:4   231       WRITELN(LINE);
1   1:4   247       (*$I+*)
1   1:4   247       (*$R+*)
1   1:4   247       INSTFILE^.LINE[I]:=LINE;
1   1:3   263       END;
1   1:2   273   SEEK(INSTFILE,J);
1   1:2   283   PUT(INSTFILE);
1   1:1   290   UNTIL EOF(ORIGINST);
1   1:0   300 END.
```

BLOCKINSTR—Blocks the instruction data set for use with the APM package. By using a blocked data set, processing is speeded.

```
1    1:D     1 (*$L PRINTER: *)
1    1:D     1 (*Program to take text help file and convert it to blocked help file*)
1    1:D     1 (*  After editing file, X BLOCKHELP.  At the pause, place this disk in*)
1    1:D     1 (*  Drive #1 and place the APM UTIL disk in Drive #2.  Press any key.*)
1    1:D     1 (*  the filenames are: #4:Help1.text, #4:Help2.text, #4:Help3.text  or,*)
1    1:D     1 (*  you may use the BRIEFHELP files instead*)
1    1:D     1 (*  Within a few minutes, files will be blocked.*)
1    1:D     1 (*Note:
1    1:D     1     Each frame of text must be exactly 10 lines long in the text file!*)
1    1:D     1 (*Ronald G. Shapiro                V2.0                    10/25/82*)
1    1:D     1 PROGRAM BLOCKHELP;
1    1:D     3
1    1:D     3 TYPE
1    1:D     3 HELPRFILE=RECORD
1    1:D     3 LINE:ARRAY[1..10] OF STRING[80];
1    1:D     3 END;
1    1:D     3
1    1:D     3 VAR
1    1:D     3 HELPFILE:FILE OF HELPRFILE;
1    1:D   713 ORIGHELP:TEXT;
1    1:D  1014 I,J,K,L,M,N:INTEGER;
1    1:D  1020 LINE:STRING[80];
1    1:D  1061 FILENAME: STRING[80];
1    1:D  1102 A:CHAR;
1    1:D  1103
1    1:0     0 BEGIN
1    1:1     0 WRITELN('Pause--set up disks--then anykey (return)');
1    1:1    87 READLN;
1    1:1    95 J:=0;
1    1:1    99 REWRITE(HELPFILE,'#5:HELP');
1    1:1   118 CLOSE(HELPFILE,PURGE);
1    1:1   126 REWRITE(HELPFILE,'#5:HELP');
1    1:1   145 SEEK(HELPFILE,1);
1    1:1   153 PUT(HELPFILE);
1    1:1   160 CLOSE(HELPFILE,LOCK);
1    1:1   168 RESET(HELPFILE,'#5:HELP');
1    1:1   187 REPEAT
1    1:2   187   WRITE('Input Filename (esc if done): ');
1    1:2   229   readln(filename);
1    1:2   249   IF (ORD(FILENAME[1])=27) THEN
```

BLOCKHELP--Blocks the HELP (and BRIEFHELP) data set for use with the APM package. By using a blocked data set, processing is speeded.

```
1    1:3   259      EXIT(PROGRAM);
1    1:2   263    RESET(ORIGHELP,filename);
1    1:2   276
1    1:2   276    REPEAT
1    1:3   276      J:=J+1;
1    1:3   284      FOR I:=1 TO 10 DO
1    1:4   301        BEGIN
1    1:4   301          (*$R-*)
1    1:4   301          (*$I-*)
1    1:5   301          READLN(ORIGHELP,LINE);
1    1:5   317          M:=0;
1    1:5   321          FOR K:=80 DOWNTO 1 DO
1    1:6   338            BEGIN
1    1:7   338              A:=LINE[K];
1    1:7   348              IF (ORD(A)<29)OR(ORD(A)>127) THEN
1    1:8   361                BEGIN
1    1:9   361                  DELETE(LINE,K,1);
1    1:8   371                  END;
1    1:6   371              END;
1    1:5   381          WRITELN(LINE);
1    1:5   397          (*$I+*)
1    1:5   397          (*$R+*)
1    1:5   397          IF J>0 THEN HELPFILE^.LINE[I]:=LINE;
1    1:4   420          END;
1    1:3   430      IF J>0 THEN
1    1:4   437      SEEK(HELPFILE,J);
1    1:3   447      IF J>0 THEN
1    1:4   454      PUT(HELPFILE);
1    1:2   461      UNTIL EOF(ORIGHELP);
1    1:2   471    close(orighelp);
1    1:1   480    until (ord(filename[1])=27);
1    1:0   490 END.
```

See previous page for program description.

```
1     1:D     1 (*$L PRINTER:*)
1     1:D     1 (*RUNNING THE FOLLOWING PROGRAM MODIFIES THE SYSTEM.APPLE FILE FOR USE W/VIDEX*)
1     1:D     1 PROGRAM VIDPATCH;
1     1:D     3 VAR BUF:PACKED ARRAY[0..31,0..511] OF 0..255;
1     1:D  8195 F:FILE;
1     1:D  8235 I:INTEGER;
1     1:0     0 BEGIN
1     1:1     0   RESET(F,'#4:SYSTEM.APPLE');
1     1:1    43   I:=BLOCKREAD(F,BUF,32);
1     1:1    65   CLOSE(F);
1     1:1    74
1     1:1    74   BUF[3,389]:=160;
1     1:1   100   BUF[3,390]:=48;
1     1:1   124   BUF[3,394]:=60;
1     1:1   148   BUF[3,455]:=173;
1     1:1   174   BUF[3,456]:=0;
1     1:1   198   BUF[3,457]:=192;
1     1:1   224   BUF[3,458]:=16;
1     1:1   248   BUF[3,459]:=29;
1     1:1   272   BUF[3,460]:=32;
1     1:1   296   BUF[3,461]:=24;
1     1:1   320   BUF[3,462]:=218;
1     1:1   346   BUF[3,463]:=234;
1     1:1   372   BUF[4,207]:=3;
1     1:1   396   RESET(F,'#4:SYSTEM.APPLE');
1     1:1   424   I:=BLOCKWRITE(F,BUF,32);
1     1:1   446   CLOSE(F);
1     1:0   455 END.
```

VIDPATCH (written by VIDEX) updates the SYSTEM.APPLE program for use with the VIDEX board. This program must be run once with each SYSTEM.APPLE file.