



NAVAL MEDICAL RESEARCH INSTITUTE

Bethesda, MD 20889-5055

NMRI 91-26

March 1991

AD-A236 482



COMPUTER-BASED TECHNIQUES FOR COLLECTION OF PULMONARY FUNCTION VARIABLES DURING REST AND EXERCISE

N. A. S. Taylor
J. R. Clarke

Naval Medical Research
and Development Command
Bethesda, Maryland 20889-5044

Department of the Navy
Naval Medical Command
Washington, DC 20372-5210



DTIC
SEARCHED
SERIALIZED
INDEXED
FILED
MAR 1991
A-1

Approved for public release;
distribution is unlimited

91-01163



NOTES

The opinions and assertions contained herein are the private ones of the writer and are not to be construed as official or reflecting the views of the naval service at large.

When U. S. Government drawings, specifications, or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Please do not request copies of this report from the Naval Medical Research Institute. Additional copies may be purchased from:

National Technical Information Service
5285 Port Royal Road
Springfield, Virginia 22161

Federal Government agencies and their contractors registered with the Defense Technical Information Center should direct requests for copies of this report to:

Defense Technical Information Center
Cameron Station
Alexandria, Virginia 22304-6145

TECHNICAL REVIEW AND APPROVAL

NMRI 91-26

The experiments reported herein were conducted according to the principles set forth in the current edition of the "Guide for the Care and Use of Laboratory Animals," Institute of Laboratory Animal Resources, National Research Council.

This technical report has been reviewed by the NMRI scientific and public affairs staff and is approved for publication. It is releasable to the National Technical Information Service where it will be available to the general public, including foreign nations.

LARRY W. LAUGHLIN
CAPT, MC, USN
Commanding Officer
Naval Medical Research Institute

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCL			1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited	
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE				
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NMRI 91-26			5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION Naval Medical Research Institute		6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Naval Medical Command	
6c. ADDRESS (City, State, and ZIP Code) 8901 Wisconsin Avenue Bethesda, MD 20814-5055			7b. ADDRESS (City, State, and ZIP Code) Department of the Navy Washington, DC 20372-5120	
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Naval Medical Research & Development Command		8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS (City, State, and ZIP Code) 8901 Wisconsin Avenue Bethesda, MD 20814-5044			10. SOURCE OF FUNDING NUMBERS	
			PROGRAM ELEMENT NO. 63713N	PROJECT NO. M0099
			TASK NO. .01B	WORK UNIT ACCESSION NO. 1005
11. TITLE (Include Security Classification) (U) COMPUTER-BASED TECHNIQUES FOR COLLECTION OF PULMONARY FUNCTION VARIABLES DURING REST AND EXERCISE				
12. PERSONAL AUTHOR(S) Nigel A.S. Taylor and John R. Clarke				
13a. TYPE OF REPORT Technical Report		13b. TIME COVERED FROM 01/90 TO 12/90		14. DATE OF REPORT (Year, Month, Day) 1991 March
15. PAGE COUNT 76				
16. SUPPLEMENTARY NOTATION				
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) lung volumes, flow-volume loops, respiratory timing, FEV _{1.0} , maximal voluntary ventilation, post-inspiratory and post-expiratory pauses, peak expiratory flow	
FIELD	GROUP	SUB-GROUP		
19. ABSTRACT (Continue on reverse if necessary and identify by block number) Pulmonary function tests are routinely included as part of the experimental protocol for both immersion and barophysiology experiments. Typically, these tests are performed using standard spirometry, and as such suffer from time consuming data analysis and experimental error associated with such analysis. In many cases the combined effects of these limitations has meant that some valuable data often fails to come out in the analysis. One solution to this situation is to transfer these tasks to the computer. We have developed a computer-based system for both data collection and analysis, which is faster, more sensitive to minor physiological perturbations, and more precise than standard spirometry. The apparatus and programming algorithms for this system are presented in this report so that other laboratories might be able to similarly improve the methods for collection of pulmonary function data.				
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Regina E. Hunt, Command Editor			22b. TELEPHONE (Include Area Code) (202) 295-0198	22c. OFFICE SYMBOL SD/RSD/NMRI

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

TABLE OF CONTENTS

ABSTRACT	i
ACKNOWLEDGEMENTS	iv
INTRODUCTION	1
METHODS	3
Apparatus	3
Data acquisition	4
Calibration	6
Rest and exercise respiratory patterns	7
Lung volume measures	7
Maximal voluntary ventilation	10
Flow-volume loops and derived variables	10
REFERENCES	12
FIGURE 1	9
FIGURE 2	11
APPENDIX 1	13
APPENDIX 2	19
Part I	19
Part II	31
APPENDIX 3	34
APPENDIX 4	39
APPENDIX 5	40
APPENDIX 6	41
APPENDIX 7	44
APPENDIX 8	57
GLOSSARY OF TERMS AND ABBREVIATIONS	71

ACKNOWLEDGEMENTS

Naval Medical Research and Development Command, Research Task No. M0099.01B.1005. This work was completed while the author held a National Research Council Research Associateship at the Naval Medical Research Institute. The views, opinions and/or observations should not be construed as or reflecting the views, policy, or decisions of the Navy Department or the naval service at large, unless so designated by other official documentation.

INTRODUCTION

Pulmonary Function Tests (PFTs) are routinely included in experimental protocols involving hyper- and hypobaric excursions. Unfortunately, the full potential of those tests is often not realized due to the limited capabilities of conventional data recording technology. A potential improvement came in the last decade as microcomputers became available for clinical PFTs. However, microcomputer use in the research setting has been limited due to the perceived "black box" (i.e. undocumented) nature of many such devices. Nevertheless, the American Thoracic Society (ATS) stated the advantages of computer acquisition of pulmonary function data (1) as follows:

- (1) Complete automation of a procedure may result in significantly reduced time and cost, and in increased accuracy;
- (2) Assurance that standardized procedures are followed;
- (3) Significant reduction in major measurement errors;
- (4) Storage and retrieval of information quickly and efficiently;
- (5) Implementation of automated calibration and system check procedures within the instrument;
- (6) Standardized and consistent interpretation of results."

A recent example illustrates the power in these statements. Respiratory flow data acquired by a computer on a dive to 450 m revealed oscillations that were suspiciously similar to the frequency signature of HPNS tremor (2). Had a conventional mechanical spirometer been used, the oscillations might have been dismissed as mechanical artifacts. However, the accuracy and frequency resolution of the computer lent credence to the hypothesis that the observed oscillations represented a previously unknown action of the High Pressure Nervous Syndrome.

The ATS suggests comprehensive quality assurance guidelines for the use of digital computers in the laboratory (3). Those guidelines were referred to in the ATS standards for spirometry promulgated in 1987 (4). According to the standards, users of PFT software should possess (and hopefully understand) source code listings (computer instructions written in a high level language such as Basic, C, or Fortran). Unfortunately, commercial software is typically proprietary, meaning source code is not

available. Even when code is presented, it is frequently poorly documented. These restrictions are undesirable in the clinical setting, but are intolerable in scientific and engineering tasks.

The purpose of this report is to describe the hardware installation and source code for a Pulmonary Function data acquisition system that has proven useful in the hyperbaric research laboratory. It illustrates how computers can economically collect, analyze and display data from a variety of pulmonary function tests (Table 1). [Readers interested in computer aided measurement of dynamic, quasi-static and static pulmonary compliance are directed to Taylor (5).]

METHODS

Apparatus

The current laboratory uses a PC(AT) clone (Model Z-248, Zenith Data Systems, St. Joseph, MI) although any IBM compatible system would suffice. The critical component is the analogue-to-digital (A/D) convertor, as this determines both the maximal sampling rate and the ease of software development. The MetraByte DAS-16F A/D board (Metrabyte Corp., Taunton, MA) has a maximal sampling rate of 100 kHz, can take single ended input from 16 channels (or eight differential analogue inputs), permits both foreground and background data collection, and is fully programmable using either commercially available or laboratory-developed software. Flow was measured by a Hans Rudolph pneumotachograph (Model 4813, Hans Rudolph, Inc., Kansas City, MO), heated to 37 °C and coupled to a Validyne differential pressure transducer (MP45 $\pm$ 0.196 kPa ($\pm$ 2 cmH₂O), Validyne Engineering, Northridge, CA). Mouth pressure was obtained using a second Validyne differential pressure transducer (MP45 $\pm$ 9.803 kPa ($\pm$ 100 cmH₂O)) connected to the airway between the mouth and the pneumotachograph. Unfiltered output from the flow and pressure transducers was amplified using Validyne CD18 and CD19 carrier demodulators, respectively.

The frequency content of flow generated during the Forced Vital Capacity maneuver extends to at least 12 Hz (6). Digital sampling theory (7) requires sampling to occur at frequencies at least double that in the sampled signal. Accordingly, data from both inputs was sampled at 50 Hz. Sampling at even 100 Hz is not uncommon.

[Validyne is now manufacturing an interface card (UPC607) which serves both data acquisition and signal conditioning needs, removing the need for separate signal demodulators].

Before the DAS-16F board can be used it must first be correctly installed into the computer. This installation involves both hardware and software. Physical installation requires the setting of four D.I.P. switches, and then the correct positioning of the board. The user must correctly position the switches for: (a) channel configuration (16 or 8 channel use); (b) unipolar (positive only) or bipolar (positive and negative) voltage inputs; (c) direct memory access level (select 1 for systems with hard-disk and 3 for

floppy drive systems); and (d) gain, which operates interactively with switch (b) to set the input voltage range. Software installation is performed using INSTALL.EXE (software provided with the board), which sets the desired base address for the board within your system. Once installed the card should be calibrated using supplied software (CALF.EXE or CALBAS; see DAS-16F Manual).

Data acquisition

Appendix 1 contains a generic subroutine for the collection of data in either foreground mode (all other computer functions temporarily suspended) or background mode (for simultaneous data collection and processing). Data acquisition subroutines are presented before calibration subroutines, since to calibrate using the computer you must first be able to collect data. All computer programs were written using Microsoft Fortran77, and are presented in their smallest parts (subroutines).

The information below provides a brief overview of the commands issued by the Fortran software to the MetraByte board. Before data collection, four initializations (operation modes) must be completed:

- (1) Mode 0: This mode is used to initialize the A/D board by first testing its physical presence, then setting an input/output (base) address (variable 1), setting the interrupt level (variable 2), and setting the direct memory access (D.M.A.) level (variable 3). The D.M.A. level is set both physically on the board and within the software (see DAS-16F Manual).
- (2) Mode 1: This mode sets the lower (variable 1) and upper ranges (variable 2) of the channels to be scanned.
- (3) Mode 17: This mode sets the programmable timer rate and therefore determines the sampling rate, and requires two variables: (a) divide data for counter 2 (variable 1), and (b) divide data for counter 1 (variable 2). These counters of the 8254 counter timer produce the output pulse rate for triggering the A/D conversions. The sampling frequency is derived from the following equation:

Frequency = $1,000,000 / [(variable\ 1 / number\ of\ channels) * variable\ 2]$ However, the user is responsible for doing this division (a point not in the manual), so set the size of variables 1 and 2 with this in mind. For example, to set a sampling rate of 50 Hz for 1 channel variable 1 is set to 1000 and variable 2 is set to 20. To collect 2 channels at 50 Hz, leave variable 2 at 20 and set variable 1 to 500 (i.e. $1000/2$).

- (4) Mode 19: This last initialization mode allows for the use of an external trigger (positive input voltage) to initiate the start of A/D conversions. This is an extremely useful feature which permits remote operation by either the experimenter or the subject. It effectively suspends the computer program until a pre-determined input voltage is received. Three variables are required to operate this mode:
- (a) channel number for the analogue input voltage,
 - (b) the magnitude of the trigger voltage (bit equivalent), and
 - (c) the slope of the input signal (i.e. positive or negative).

Data acquisition is performed using one of several modes (see DAS-16F Manual), however Modes 4 (foreground) and 5 (background) are sufficient for most applications:

- (5) Mode 4: Performs A/D conversions in the foreground according to the channels ranges and sampling rates determined above. Three variables values are required: (a) the number of conversions (word count) to be performed (e.g. for 2 channels collecting data at 50 Hz for 10 seconds, word count = $2 * 50 * 10$); (b) an array (previously initialized and dimensioned within the Fortran code) into which the input shall be loaded, and the position within the array from which to start the storage (this is usually zero for a single

dimension array or '[1,0]' for an array with several levels); and (c) the source of the trigger to start the A/D conversions (set this to 1 to use the previously programmed interval timer, Mode 17).

- (6) Mode 5: Performs A/D conversions in the background according to the channels ranges and sampling rates determined above. Four input variables are required: (a) the number of conversions (word count) to be performed (as above); (b) the segment (16 bit) of the memory to receive data (you need to seek technical help here as this location is dependent upon the memory allocations already assigned within your system; we use #5000 and #6000 {Fortran}, or &H5000 and &H6000 {Basic}); (c) the source of the trigger input to start the A/D conversions (as above); and (d) a command telling the computer to carry out this sequence once (single cycle) or to operate continuously (until stopped using Mode 7 - see DAS-16F Manual).

Calibration

Calibration of the pneumotachograph must be performed under two conditions; zero flow, and using a standard volume (e.g. 3 liter) passed through the pneumotachograph at flow rates appropriate to a given experiment. Using an analysis routine, flow is integrated with respect to time, providing both volume and flow calibration. This method is superior to electronic integration, since the latter technique invariably produces an electronic drift in the baseline, making data interpretation impossible.

Mouth pressure is often of interest, especially during exercise protocols. Prior to calibration the linearity of the mouth pressure transducer must be verified. Subsequent calibration may then be performed using a two point calibration over the positive range of anticipated mouth pressures. From linear regression, the pressure data yielded both the zero offset and slope for the calibration. These

values were saved and used during data analysis. The calibration subroutines for the pneumotachograph and mouth pressure transducer are contained in Appendix 2.

Rest and exercise respiratory patterns

Respiratory pattern data (which includes V_T , V_E , and respiratory timing variables) were derived from the same routines that provided lung volume data. The difference lay in the breathing pattern used by the subject, rather than within the computer program. The example provided in Appendix 3 is from a program written to analyze data collected at 450 m (changes in ambient pressure are automatically taken into account by these procedures). Subjects exercised for 25 min while exposed to a moderate external airway resistance. Appendix 8 contains the entire program. This program also included the measurement of mouth pressure (including the determination of peak-to-peak values and inspiratory to expiratory ratios) and the integration of mouth pressure into trajectory data for a flow-volume-pressure three-dimensional surface (8).

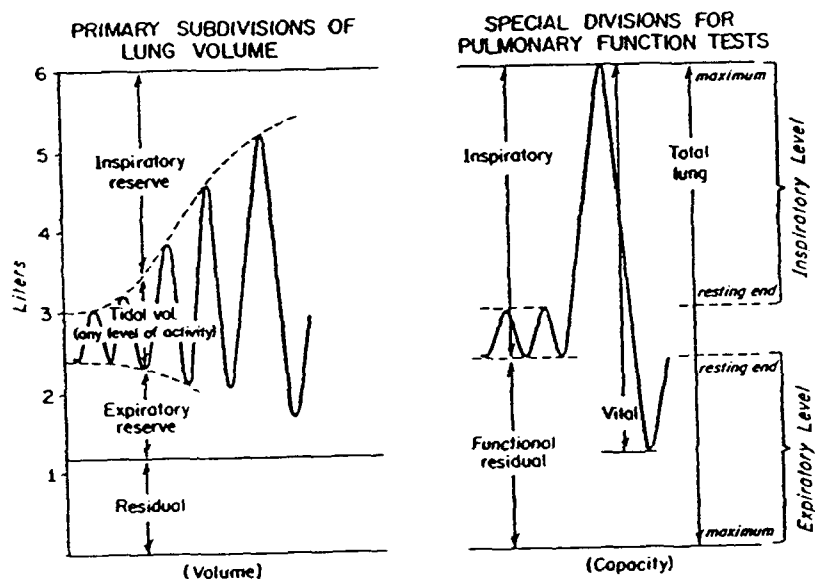
Lung volume measures

The measurement of lung volumes (excluding RV) relies more upon the skills of the experimenter than the art of the programmer. Subjects must be trained to provide reproducible VC maneuvers through the attainment of consistent RV and TLC end points. Once this is accomplished the experimenter can, with a single maneuver, obtain VC, ERV, IC, and IRV (Figure 1). Following an exhalation to RV, the subject slowly inspired to maximum lung capacity (TLC), followed by an expiration to RV and a normal slow inspiration. The subject then took normal breaths (giving V_T , ERV, and by subtraction, IC). From this point the subject inspired to TLC again. By subtracting from IC the V_T obtained using Appendix 3 (since the V_T in this maneuver is often falsely large), IRV is derived.

Subroutines necessary for measuring the above lung volumes are contained in Appendix 4. Note that the programming code is the same for each lung volume; the changes being only in the maneuver performed by the subject. These subroutines, and those included below (except the flow-volume subroutines), perform analysis on complete respiratory cycles only. This is essential, since during exercise subjects do not commence inspiration at the same time as the computer starts to collect data,

and rarely terminate exhalation immediately prior to the end of data collection. Therefore, within the initial analysis subroutine (Appendix 3) is a 'paragraph' which searches the flow data to find the first commencement of inspiration. This is found when the flow data crosses the predetermined basal flow threshold in the inspiratory direction. The subroutine then counts the number of inspirations (counter I) and expirations (counter E). To ensure these are always equal, the last inspiration is omitted, and the output is always written for I-1 inspirations and expirations.

FIGURE 1



Standard lung volumes and capacities.

Maximal voluntary ventilation

The MVV maneuver (Appendix 5) is also a hybrid of the breathing pattern subroutines (Appendix 3). For the MVV, all the V_T 's are summed then divided by the time taken to complete the maneuver. This time is controlled by the experimenter so that it equals the duration of interest (i.e. 12 or 15 seconds). As with the above subroutines, only complete cycles are analyzed, so the time to complete the maneuver (as determined by the software) will often not equal the time used during the experiment. This difference is minimal, and results in greater precision in the determination of MVV than is possible using a spirometer.

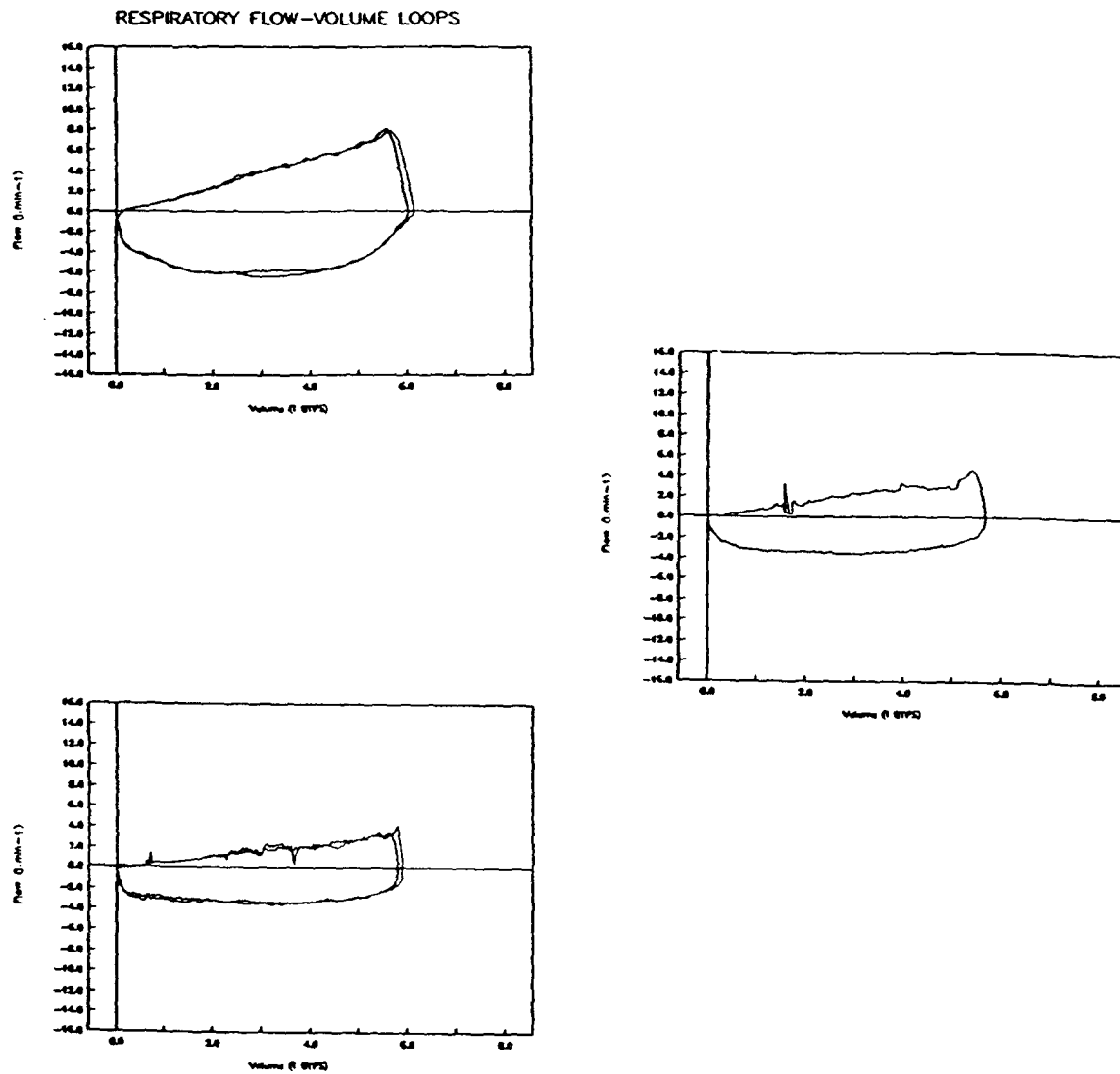
Flow-volume loops and derived variables

For this maneuver (Appendix 6) subject training was again vital to ensure that subjects obtained true maximal flows and reproducible end points at RV and TLC. The amplifier gain setting was adjusted so that voltages during peak flows were as close as possible to the ± 5 Volt input capacity of the A/D board.

Since data sampling was at 50 Hz, flow and instantaneous lung volumes were derived at 20 ms intervals over the entire volume excursion. These data were saved as an ASCII file for importation into suitable graphics packages (e.g. Lotus123 (Figure 2) or Sigmaplot [see: Taylor 1990]).

The flow data obtained from these maneuvers was also used to obtain the $FEV_{1.0}$ for each expiration. The ASCII flow data was searched manually to detect PEFR and MEFR for various percentages of the VC. While algorithms for these latter computations may be written and incorporated into the analysis subroutines, it is often easier to take these values by hand directly from the ASCII files. If noise occurred in the flow signal at the volume of interest, computer analysis could be confounded.

FIGURE 2



Flow-volume loops obtained during trials conducted at the surface,
500 fsw and at 1000 fsw. All plots are for the same subject.

REFERENCES

1. Gardner RM, Clausen JL, Cotton DJ, Crapo RO, Epler GR, Hankinson JL, and Johnson RL, Jr. Computer guidelines for pulmonary laboratories. American Review of Respiratory Disease, Vol. 134, No. 3, pp. 628-629, 1986.
2. Fraser I.M., and Flook V., "HPNS tremor on flow-volume curves at 450 meters." In: Sterk W. and Geeraedts L (eds.). Proceedings of the European Undersea Biomedical Research Society. Amsterdam, Netherlands. August 11-18, 1990, pp. 75-81.
3. ATS Statement. Quality assurance in pulmonary function laboratories. American Review of Respiratory Disease, Vol. 134, pp. 625-627, 1986.
4. Gardner RM, Hankinson JL, Clausen JL, Crapo RO, Johnson RL, Jr, and Epler GR. Standardization of spirometry - 1987 update. American Review of Respiratory Disease, Vol 136, pp. 1285-1298, 1987.
5. Taylor N.A.S., "Computer integrated measurement of respiratory compliance." Computers in Biology and Medicine, Vol. 20, No. 1, pp. 55-64, 1990.
6. Lemen RJ, Gerdes CB, Wegmann MJ, and Perrin KJ. Frequency spectra of flow and volume events for forced vital capacity. Journal of Applied Physiology, Vol. 53, pp. 977-984, 1982.
7. Jerri AJ. The Shannon sampling theorem-its various extensions and applications; a tutorial review. Proceedings of the IEEE, Vol. 65, pp. 1565-1596, 1977.
8. Marshall RN, Mazur SM, and Taylor NAS. Three dimensional surfaces for human muscle kinetics. European Journal of Applied Physiology, 1990, In press.

Data acquisition subroutines.

13

```

c
c Initialization of the MetraByte A/D board: see text for description
c
    IMODE = 0
    PARAM(1) = #300
    PARAM(2) = 2
    PARAM(3) = 3
    RCODE = DASH16(IMODE, PARAM)

c
c Setting the channel scanning limits for 3 channels, starting at 2 and
c ending at 4.
c
    IMODE = 1
    PARAM(1) = 2
    PARAM(2) = 4
    RCODE = DASH16(IMODE, PARAM)

c
c Setting the sampling rate for three channels at 50 Hz
c (i.e. 1,000,000/[(333=1000/3)*20])

    IMODE = 17
    PARAM(1) = 333
    PARAM(2) = 20
    RCODE = DASH16(IMODE, PARAM)
    WRITE(*,12)
12  FORMAT(///'          PRESS THE TRIGGER TO'/
    *  ' ***** ACTIVATE DATA COLLECTION *****'/////)

c
c Using an analog trigger on channel 2 to activate data collection.
c Program pauses until an positive voltage greater than 300 (binary
c units) is detected. This is a very useful Mode that can be used at
c any point in the program. Unlike large machines, the PC is unable
c to check hardware status while completing other tasks, so this Mode
c can also be used to terminate operations as well as to initiate.
c
    IMODE = 19
    PARAM(1) = 2
    PARAM(2) = 300
    PARAM(3) = 0
    RCODE = DASH16(IMODE, PARAM)

c
    WRITE(*,141) BELL
    WRITE(*,13)
13  FORMAT(///
    *  ' ...../
    *  ' *** COLLECTION STARTED ***/
    *  ' .....'/////)

c
c Background data collection using Mode 5. See test for description.
c Note that the Variable/Arrays for this CALL have been altered from
c previous modes. This is because they need to be unique to this CALL
c since it will be operating in the background and will be confused if

```

c these values are altered during operation.

c

```
LMODE = 5
LPARAM(1) = 9000
LPARAM(2) = #5000
LPARAM(3) = 1
LPARAM(4) = 0
LCODE = DASH16(LMODE, LPARAM)
ZZ = ZZ + 1
```

c

7777 CONTINUE

c

c

c Mode 8 is used to monitor the status of the background data collection.
c In this example we wish 1 full minute of data collection to be transferred
c into an array from memory (3 chans * 50 Hz * 60 sec = 9000). Mode 8 keeps
c the program sitting till the 'word count' equals 9000. We could use this
c Mode to transfer data in smaller segments for real-time processing. This is
c the real advantage that Mode 5 provides.

c

```
3330 JMODE = 8
      JCODE = DASH16(JMODE, VAR)
      IF (VAR(3).GE.9000) GOTO 3001
      GOTO 3330
3001 WRITE(*,3900) ZZ
3900 FORMAT(' + MINUTE ',I2,' COMPLETE:')
```

c

c Here Mode 4 is used to monitor channel 2 for a positive input from
c the trigger (i.e. greater than binary value of 500). Mode 1 is not
c re-defined, instead all 3 channels are monitored. However, the 'word
c count' is set to 1 to remove the input from the other channels.
c Note: this only works because the trigger is connected to channel 2.
c If trigger input is detected the program jumps to Statement 4000 to
c save the data, then to Statement 8 to terminate data collection.

c

```
IMODE = 4
PARAM(1) = 1
PARAM(2) = OFFADR(CHECK)
PARAM(3) = 1
RCODE = DASH16(IMODE, PARAM)
IF (CHECK.GT.500) GOTO 4000
IF (TCOUNT.GE.TIME) GOTO 4000
```

c Mode 5 is activated again immediately after the trigger check. This
c results in minimal data loss (milliseconds).

c

```
LMODE = 5
LPARAM(1) = 9000
```

```

        LPARAM(2) = #6000
        LPARAM(3) = 1
        LPARAM(4) = 0
        LCODE = DASH16(LMODE, LPARAM)
        ZZ = ZZ + 1
c
c Mode 9 is then used to transfer data from memory to a previously
c initialized and dimensioned array. This occurs in the foreground
c while the data acquisition subroutines are operating in background.
c Five input Variables are required for Mode 9:
c (1) 'word count', 9000 = 3 chans * 50 Hz * 60 seconds
c (2) memory address as per Mode 5 declaration
c (3) the conversion number to start the transfer from
c (4) the array name and location within the array (e.g. point 0 of level 1)
c (5) channel array data: use 0 if not required.
c
4000    KMODE = 9
        ELEM(1) = 9000
        ELEM(2) = #5000
        ELEM(3) = 0
        ELEM(4) = OFFADR(DATA1(1,0))
        ELEM(5) = 0
        KCODE = DASH16(KMODE, ELEM)
c
c Now while Mode 5 is still operating the data are written to a
c file on the disk in ASCII code. This file is not closed, but is
c continually added to with data from each usuccessive minute.
c
        DO 3002 I=1,(9000/3)
        WRITE(3,4949) (DATA1(J,I),J=2,3)
3002    CONTINUE
4949    FORMAT(2I6)
c
        TCOUNT = TCOUNT + 1
        IF (CHECK.GT.500) GOTO 8
        IF (TCOUNT.GE.TIME) GOTO 8
c
3331    JMODE = 8
        JCODE = DASH16(JMODE, VAR)
        IF (VAR(3).GE.9000) GOTO 3004
        GOTO 3331
3004    WRITE(*,3900) ZZ
c
        IMODE = 4
        PARAM(1) = 1
        PARAM(2) = OFFADR(CHECK)
        PARAM(3) = 1
        RCODE = DASH16(IMODE, PARAM)
c
        IF (CHECK.GT.500) GOTO 4001
        IF (TCOUNT.GE.TIME) GOTO 4001
c

```

```

      LMODE = 5
      LPARAM(1) = 9000
      LPARAM(2) = #5000
      LPARAM(3) = 1
      LPARAM(4) = 0
      LCODE = DASH16(LMODE, LPARAM)
      ZZ = ZZ + 1
c
4001  KMODE = 9
      ELEM(1) = 9000
      ELEM(2) = #6000
      ELEM(3) = 0
      ELEM(4) = OFFADR(DATA2(1,0))
      ELEM(5) = 0
      KCODE = DASH16(KMODE, ELEM)
c
      DO 3005 I=1,(9000/3)
      WRITE(3,4949) (DATA2(J,I),J=2,3)
3005  CONTINUE
c
      TCOUNT = TCOUNT + 1
      IF (CHECK.GT.500) GOTO 8
      IF (TCOUNT.GE.TIME) GOTO 8
c
      GOTO 7777
c
8      WRITE(*,141) BELL,BELL,BELL,BELL,BELL
141    FORMAT('  'A1,'  'A1,'  'A1,'  'A1,'  'A1)
      WRITE(*,14)
14     FORMAT(//
*      ' ...../
*      ' *** COLLECTION FINISHED ***/
*      ' ...../)
20     FORMAT(I1)
c
      CLOSE(3)
c

      WRITE(*,2101) FILENAME
2101  FORMAT(/////////
*      '   DATA FROM: 'A20,' SAVED:'''/
*      ' ...../
*      '  RECORD EXPERIMENT TIME!''/
*      ' .....//
*      '   WHAT DO YOU WISH TO DO NEXT?'/
*      '   1 RETURN = COLLECT MORE DATA'/
*      '   9 RETURN = TERMINATE PROGRAM'/)
      READ(*,20) ANS2
      IF (ANS2.EQ.1) GOTO 100

```

C

9191 FORMAT(//////////)

STOP

[illegible]

Calibration subroutines.

[illegible][illegible]

```

INTEGER ZERO,GAIN,PEAKMODE,ZCHAN,HIGH,MM,M,B,D,E,F,G,H,J,K,IMODE
INTEGER ZER,HIG,COUNT,GAIN2,BASE,BAS,ICHAN,BIG,BOG,PARAM,SAVE
INTEGER MEANNOFLOW,LITER,NOFLOW,GAIN3,ANS,DASH16,OFFADR,RCODE
REAL MEANZERO,MEANHIGH,PRESSURE,REGCOEFF,CONSTANT
REAL V,W,X,Y,Z
REAL RZERO,RHIGH,A,C,RBASE,RBIG,PRESS2
CHARACTER*40 FILENAME,FILE2,FILE3,FILE4
CHARACTER BELL
DIMENSION ICHAN(2),ZERO(11),HIGH(11),RZERO(11),RHIGH(11)
DIMENSION BASE(11),RBASE(11),BIG(11),RBIG(11),PARAM(4)
DIMENSION NOFLOW(101),LITER(2000)

```

CCCCCCCCCCCCCCCCCC

19


```

311  CONTINUE
c
    WRITE(*,4)
4    FORMAT(/' ***** CALIBRATION VALUES COLLECTED *****'/)
    WRITE(*,4000) BELL
c
c Calculating mean zero pressure.
c
    MEANZERO = A / 10.0
    WRITE(*,41) (RZERO(J),J=2,11),MEANZERO
41   FORMAT(2(5(2X,F9.2)//),' MEAN ZERO PRESSURE = ',F9.2//)
    WRITE(*,42)
42   FORMAT(' CHECK BASAL VALUES ARE ALL ABOUT EQUAL.')
c
    WRITE(*,43)
43   FORMAT(//' WHAT DO YOU WISH TO DO NEXT?'/
*   ' 1 RETURN = CONTINUE TO NEXT STAGE, '/
*   ' 6 RETURN = REPEAT & ERASE BASAL CALIBRATION, '/
*   ' 9 RETURN = END PROGRAM. '/)
    READ(*,20) B
    IF (B.EQ.1) CONTINUE
    IF (B.EQ.6) GOTO 10
    IF (B.EQ.9) GOTO 9
c
50   WRITE(*,5)
5    FORMAT(///' APPLY SUITABLY LARGE PRESSURE TO THE TRANSDUCER. '/
*   ' THIS PRESSURE SHOULD BE GREATER THAN THE PRESSURE '/
*   ' YOU ANTICIPATE DURING THE EXPERIMENT. '//////)
c
    IMODE = 0
    PARAM(1) = #300
    PARAM(2) = 2
    PARAM(3) = 3
    RCODE = DASH16(IMODE, PARAM)
c
    IMODE = 1
    PARAM(1) = 3
    PARAM(2) = 3
    RCODE = DASH16(IMODE, PARAM)
c
    MODE = 17
    PARAM(1) = 1000
    PARAM(2) = 20
    RCODE = DASH16(IMODE, PARAM)
c
    WRITE(*,51)
51   FORMAT(' **** PRESS TRIGGER TO COLLECT CALIBRATION ****'//)
c
    IMODE = 19
    PARAM(1) = 2
    PARAM(2) = 300
    PARAM(3) = 0

```

```

RCODE = DASH16(IMODE, PARAM)
WRITE(*,31)
WRITE(*,4000) BELL
c
IMODE = 4
PARAM(1) = 12
PARAM(2) = OFFADR(HIGH(0))
PARAM(3) = 1
RCODE = DASH16(IMODE, PARAM)
c
C = 0.0
WRITE(*,4)
WRITE(*,4000) BELL
c
c Summing 10 peak pressures.
c
DO 511 K=2,11
HIG=HIGH(K)
RHIGH(K)=FLOAT(HIG)
C = C + RHIGH(K)
511 CONTINUE
c
c Calculating mean peak pressure.
c
MEANHIGH = C / 10.0
WRITE(*,52) (RHIGH(K),K=2,11),MEANHIGH
52 FORMAT(2(5(2X,F9.2)//),' MEAN HIGH PRESSURE = ',F9.2//)
WRITE(*,53) MEANZERO
53 FORMAT(' MEAN ZERO PRESSURE = ',F9.2//)
c
WRITE(*,54)
54 FORMAT(/' WHAT DO YOU WISH TO DO NEXT?'/
* ' 1 RETURN = LINEAR REGRESSION ANALYSIS,'/
* ' 6 RETURN = REPEAT POOR HIGH PRESSURE CALIBRATION,'/
* ' 9 RETURN = END Program.'//)
READ(*,20) D

IF (D.EQ.1) CONTINUE
IF (D.EQ.6) GOTO 50
IF (D.EQ.9) GOTO 9
c
WRITE(*,6)
6 FORMAT(/' PLEASE TYPE THE PRESSURE OF THE SECOND CALIBRATION'/
* ' USING cm.H2O (e.g. xxx.xx [F7.2]) '//)
READ(*,61) PRESSURE
61 FORMAT(F7.2)
c

c Linear regression analysis to obtain coefficients for
c equation:  $y = A + Bx$ . Where: A = y intercept,
c B = slope.
c

```



```

c      IMODE = 17
      PARAM(1) = 1000
      PARAM(2) = 20
      RCODE = DASH16(IMODE, PARAM)

c      WRITE(*,8400)
8400  FORMAT(///'      SET THE MASS SPEC OUTPUT TO ZERO:.'/
*      '      CHECK TRANSDUCER CONNECTED TO CHANNEL 5:.'/
*      '      ***** PRESS TRIGGER TO COMMENCE *****'//)

c      IMODE=19
      PARAM(1)=2
      PARAM(2)=300
      PARAM(3)=0
      RCODE = DASH16(IMODE, PARAM)

c      WRITE(*,31)
      WRITE(*,4000) BELL

c      IMODE=4
      PARAM(1)=12
      PARAM(2)=OFFADR(BASE(0))
      PARAM(3)=1
      RCODE = DASH16(IMODE, PARAM)

c      A = 0.00
      DO 841 J=2,11
      BAS=BASE(J)
      RBASE(J)=FLOAT(BAS)
      A = A + RBASE(J)
841  CONTINUE

c      WRITE(*,4)
      WRITE(*,4000) BELL
      MEANZERO = A / 10.00
      WRITE(*,41) (RBASE(J),J=2,11),MEANZERO
      WRITE(*,42)

c      WRITE(*,43)
      READ(*,20) G
      IF (G.EQ.1) CONTINUE
      IF (G.EQ.6) GOTO 84
      IF (G.EQ.9) GOTO 9

c      WRITE(*,5111)
5111  FORMAT(///' APPLY SUITABLY HIGH CO2 OUTPUT FROM MASS SPEC:.'/)

c      8420  IMODE = 0
      PARAM(1) = #300
      PARAM(2) = 2
      PARAM(3) = 3

```

```

      RCODE = DASH16(IMODE, PARAM)
c
      IMODE = 1
      PARAM(1) = 5
      PARAM(2) = 5
      RCODE = DASH16(IMODE, PARAM)
c
      IMODE = 17
      PARAM(1) = 1000
      PARAM(2) = 20
      RCODE = DASH16(IMODE, PARAM)
c
      WRITE(*,51)
c
      IMODE=19
      PARAM(1)=2
      PARAM(2)=300
      PARAM(3)=0
      RCODE = DASH16(IMODE, PARAM)
c
      WRITE(*,31)
      WRITE(*,4000) BELL
c
      IMODE=4
      PARAM(1)=12
      PARAM(2)=OFFADR(BIG(0))
      PARAM(3)=1
      RCODE = DASH16(IMODE, PARAM)
c
      WRITE(*,4)
      WRITE(*,4000) BELL
c
      C = 0.0
      DO 843 K=2,11
      BOG = BIG(K)
      RBIG(K)=FLOAT(BOG)
      C = C + RBIG(K)
843  CONTINUE
c
      MEANHIGH = C / 10.0
      WRITE(*,52) (RBIG(K),K=2,11),MEANHIGH
      WRITE(*,53) MEANZERO
c
      WRITE(*,54)
      READ(*,20) H
      IF (H.EQ.1) CONTINUE
      IF (H.EQ.6) GOTO 8420
      IF (H.EQ.9) GOTO 9
c
      WRITE(*,6111)
6111  FORMAT(///' ENTER VALUE OF HIGH % CO2: '//)
      READ(*,61) CONC2

```



```

      RCODE = DASH16(IMODE, PARAM)
c
      IMODE = 17
      PARAM(1) = 1000
      PARAM(2) = 20
      RCODE = DASH16(IMODE, PARAM)
c
      WRITE(*,860)
860  FORMAT(// 'CONNECT PNEUMOTACHOGRAPH TRANSDUCER TO CHANNEL FOUR.')
      WRITE(*,861)
861  FORMAT(/// ' *** PRESS TRIGGER TO COLLECT BASELINE DATA ***'//)
c
      IMODE=19
      PARAM(1)=2
      PARAM(2)=300
      PARAM(3)=0
      RCODE = DASH16(IMODE, PARAM)
c
      WRITE(*,31)
      WRITE(*,4000) BELL
c
      IMODE=4
      PARAM(1)=102
      PARAM(2)=OFFADR(NOFLOW(0))
      PARAM(3)=1
      RCODE = DASH16(IMODE, PARAM)
c
      WRITE(*,4000) BELL
4000  FORMAT(' ',A1)
      WRITE(*,4)
c
      A = 0
      DO 862 J=2,101
      A = A + NOFLOW(J)
862  CONTINUE
c
c Writing to the screen all the basal flow data, plus their average.
c
      MEANNOFLOW = A / 100
      WRITE(*,863) (NOFLOW(J),J=2,101),MEANNOFLOW
863  FORMAT(10(10(1X,I5)/),' MEAN BASELINE FLOW = ',I5//)
c
      WRITE(*,42)
      WRITE(*,43)
      READ(*,20) G
      IF (G.EQ.1) CONTINUE
      IF (G.EQ.6) GOTO 86
      IF (G.EQ.9) GOTO 9
c
c Saving the zero flow mean in a specific file.
c
      WRITE(*,82)

```

```

WRITE(*,4000) BELL
READ(*,73) FILE3
OPEN(3,FILE=FILE3,STATUS='NEW',FORM='FORMATTED')
WRITE(3,865) MEANNOFLOW
865  FORMAT(17)
      CLOSE(3)
c
      WRITE(*,866)
866  FORMAT(///' CALIBRATE PNEUMOTACHOGRAPH USING 1 LITER SYRINGE./'
      * '      START FLOW AS SUCTION./'
      * '      USE SIX (6) PUMPS OF VARIABLE SPEED./'
      * '      DO NOT HIT THE PLUNGER OF THE SYRINGE HARD AGAINST/'
      * '      THE ENDS OF ITS HOUSING, AS THIS PRODUCES POOR DATA.'///)
c
8651  IMODE = 0
      PARAM(1) = #300
      PARAM(2) = 2
      PARAM(3) = 3
      RCODE = DASH16(IMODE, PARAM)
c
      IMODE = 1
      PARAM(1) = 4
      PARAM(2) = 4
      RCODE = DASH16(IMODE, PARAM)
c
      IMODE = 17
      PARAM(1) = 1000
      PARAM(2) = 20
      RCODE = DASH16(IMODE, PARAM)
c
      WRITE(*,51)
c
      IMODE=19
      PARAM(1)=2
      PARAM(2)=300
      PARAM(3)=0
      RCODE = DASH16(IMODE, PARAM)
c
      WRITE(*,31)
      WRITE(*,4000) BELL
c
      IMODE=4
      PARAM(1)=2002
      PARAM(2)=OFFADR(LITER(0))
      PARAM(3)=1
      RCODE = DASH16(IMODE, PARAM)
c
      WRITE(*,4000) BELL
      WRITE(*,4)
      WRITE(*,867)
867  FORMAT(///' WHAT DO YOU WISH TO DO NOW?/'
      * '      1 RETURN = SAVE DATA,/'

```


c

9989

STOP
END

30

PART II: Integration of the calibration flow signal.

Integration subroutine used with the flow calibration data to obtain the number of computer/binary units that are equivalent to 1 litre of the volume calibration standard. This subroutine may be seen in the composite program in Appendix 7.

~~~~~

```
c Pre-requisites for this subroutine include:
c (1) calibration data filenames (NAME1 and NAME2);
c (2) a set flow limit - this is the binary equivalent of typical
c     flow encountered during the experiment (this is variable
c     between experiments, and between amplifier and their gain
c     settings);
c (3) a arbitrary volume limit - swallowing, sighing etc will produce
c     spurious volume excursions - this limit is used to prevent
c     these volumes being used during data analysis, but is also
c     used here to prevent artifacts from being included in the
c     calibration;
c (4) a volume for the calibration syringe, determined by repeated
c     water filling.
```

```

FILEBASE=NAME1
FILEFLOW=NAME2
FLOW=-500
ARBITLIM=500
SYRINGE=3.000

```

**c**  
**c Reading the zero flow calibration data from data disk.**

```

OPEN(3,FILE = FILEBASE,STATUS='OLD')
READ(3,101) ZEROFLOW
WRITE(*,102) ZEROFLOW
101 FORMAT(I7)
102 FORMAT(' BASAL FLOW (RAW UNITS) = ',I5//)
CLOSE(3)
50 FORMAT(2X,F9.2,2X,F9.2)

```

c Calculation of baselimit used to check for flow later  
c in program. This is pre-programd to suit anticipated  
c flow when volume is sucked slowly through the pneumotach to  
c give a negative signal. Transducer = Validyne  $\pm$  2.0 cmH<sub>2</sub>O.  
c The 4/100 value is used to set a 4% error level.  
c Two limits are set: (1) BASELIM for inspiration and (2) TBASELIM  
c for expiration. These serve as a zone above and below the actual  
c zero flow level within which the flow signal can oscillate before  
c triggering integration. This is very important in the analysis  
c subroutines since it minimizes the effect of artifacts without  
c compromising the integrity of the analysis.

```

c      BASE = (FLOW - ZEROFLOW) * 4 / 100
      BASELIM = BASE + ZEROFLOW
      TBASELIM = ZEROFLOW - BASE
C      WRITE(*,1345) BASE
1345  FORMAT(' VALUE OF BASE = 'I5//)
      WRITE(*,104) BASELIM
      WRITE(*,104) TBASELIM
104   FORMAT(' VALUE OF BASELIMIT = 'I5//)
c
c Reading the flow calibration data.
c
      OPEN(3,FILE=FILEFLOW,STATUS='OLD',FORM='UNFORMATTED')
      READ(3) (LITER(J),J=1,2000)
11    FORMAT(10(I7))
      CLOSE(3)
c
c Integration of flow to derive volume.
c L will represent the breath counter (i.e. # of breaths).
c J will represent the sample number currently analysed.
c
c Checking whether flow is baseline or a calibration input.
c
      M=1
      L=0
12    DO 120 J=M,2000
      IF (LITER(J).LT.BASELIM) GOTO 1200
120   CONTINUE
1200  CONTINUE
c
      IF (J.GE.2000) GOTO 129
c
c Integration of calibration flow data.
c
      VOLINSP=0.0
c
121   DO 127 T=J,2000
      IVOL=LITER(T)-ZEROFLOW
      RVOL=FLOAT(IVOL)
      VOLINSP=VOLINSP-RVOL
      IF (LITER(T).GE.BASELIM) GOTO 1270
127   CONTINUE
1270  CONTINUE
c
c Checking data for false volumes before storing volumes.
c
      IF (VOLINSP.LE.ARBITLIM) GOTO 128
      L=L+1
      CALVOL(L)=VOLINSP
128  IF (T.GE.2000) GOTO 129
      M=T
      GOTO 12

```



### Respiratory pattern subroutines.

[illegible]

**C**

3002 CONTINUE

**C**

**C**

**c Integrating flow to obtain volume (inspiration -ive).**

**C**

**C**

**C**

c

C

```

C      WRITE(*,5500) K,PNEUMO(K)
C5500  FORMAT(' IONSET AT POINT ',I5,' = ',I5)

```

```

C
      VOLINSP=0.0
      VOL2INSP=0.0
      EXVOLINSP=0.0
      EX2VOLINSP=0.0

```

c

c Since we know the signal is negative (above), we can now look for

c the next onset of inspiration. This ensures that the first inspiration

c is trully a complete breath and not partially cut by failure of the

c subject to coincide inspiration onset with the start of data collection.

c Integration of inspiratory flow (addition, ie VOLINSP-(-RVOL)).

```

c
      DO 32 T=K,MAX
      IF (PNEUMO(T).GE.BASELIM) GOTO 33
      IVOL=PNEUMO(T)-BASEFLOW
      RVOL=FLOAT(IVOL)
      VOLINSP = VOLINSP - RVOL
32     CONTINUE
33     CONTINUE

```

c

c The -RVOL converts volume to positive.

c Check for small, false volumes.

```

c
      WRITE(*,5501) VOLINSP
C5501  FORMAT(' INTEGRATED TOTAL = ',F10.2)

```

```

c
      IF (VOLINSP.LT.ARBITLIM) GOTO 3103

```

c

c Marking points of inspiration start (IONSET), & end (EONSET),

c and expiration start (STARTEXP), & end (ENDEXP).

```

c
      I=I+1
      IONSET(I)=K
      EONSET(I)=T
      DO 330 J=1,15
      VOL(J)=0.0
330    CONTINUE

```

```

c
      L=L+1

```

c

c Applying the calibration value (VOLAVE: see Appendix 2).

```

c
      VOL(L) = VOLINSP / VOLAVE

```

c

c Volume correction for regulator pressure and PLC, then BTPS

c correction (all values are in cmH2O and degC units).

c Variables: PREG = regulator pressure at the center of the diaphragm

c (if needed), PH2O = water vapour pressure at TA, TA = ambient temperature,





$$\begin{aligned} \text{AVTIDALV} &= \text{AVTIDALV} / (I-1) \\ \text{AVTINSPI} &= \text{AVTINSPI} / (I-1) \\ \text{AVTEXPI} &= \text{AVTEXPI} / (I-1) \\ \text{AVTTOTAL} &= \text{AVTTOTAL} / (I-1) \\ \text{RFREQ} &= 60.0 / \text{AVTTOTAL} \\ \text{VTTIMEINSPI} &= \text{AVTIDALV} / \text{AVTINSPI} \\ \text{MINVENT} &= \text{AVTIDALV} * \text{RFREQ} \\ \text{TINTOTAL} &= \text{AVTINSPI} / \text{AVTTOTAL} \\ \text{AVINSPIPOST} &= \text{AVINSPIPOST} / (I-1) \\ \text{AVEXPPOST} &= \text{AVEXPPOST} / (I-1) \end{aligned}$$

```

WRITE(*,440) AVTIDALV
440  FORMAT('///// AVERAGE TIDAL VOLUME = ',F12.4/)
WRITE(*,441) AVTINSPI
441  FORMAT(' AVERAGE TIME INSP = ',F12.4)
WRITE(*,442) AVTEXPI
442  FORMAT(' AVERAGE TIME EXPIR. = ',F12.4)
WRITE(*,443) AVTTOTAL
443  FORMAT(' AVERAGE TOTAL TIME = ',F12.4/)
WRITE(*,444) VTTIMEINSP
444  FORMAT(' VT/Ti = ',F12.4)
WRITE(*,446) TINTOTAL
WRITE(*,4400) RFREQ
4400  FORMAT(' BREATHING FREQUENCY = ',F12.4)
WRITE(*,445) MINVENT
445  FORMAT(' MINUTE VENT. = ',F12.4/)
446  FORMAT(' Ti/Ttot = ',F12.4/)
WRITE(*,447) AVINSPPOST
447  FORMAT(' AVERAGE POST-INSPIRATORY PAUSE = ',F12.4)
WRITE(*,448) AVEXPPOST
448  FORMAT(' AVERAGE POST-EXPIRATORY PAUSE = ',F12.4)

```

[illegible]

### Lung volume subroutines.

```

c
      WRITE(*,3337)
      DO 3339 P= 1,I
      WRITE(*,3338) P, TIDALV(P)
3339  CONTINUE
3337  FORMAT(' VOLUME #   SPIROMETRIC VOLUMES (inspiratory)')
3338  FORMAT(2X,I4,11X,F7.3)
c

```

## APPENDIX 5

### Maximal voluntary ventilation subroutine.

Calculations performed here are also dependent upon those performed in Appendix 3, since each tidal volume (TIDALV) is added for the duration of the maneuver, and the duration from the start to the end of the maneuver are required (MVVTIME). This subroutine is contained within the example program in Appendix 7.

```

c
      MVV = 0.0
c
      DO 35 P=1,(I-1)
      MVV = MVV + TIDALV(P)
35  CONTINUE
c
c The number 50.0 and 60.0 are used because the sampling rate was 50 Hz and the
c output is required in liters per minute (60.0).
c
      MVVTIME = (ENDEXP(I-1)-IONSET(1)) / 50.0 / 60.0
      MVV = MVV / MVVTIME
      WRITE(*,350) MVV
350  FORMAT(///'  MAXIMAL VOLUNTARY VENTILATION FOR THIS'/
      *      '      TRIAL = ',F12.4,' Litres BTPS'///)
c

```

### Maximal flow-volume loop subroutines.

~~~~~

60 CONTINUE

```

WRITE(*,63)
DO 610 PPP=IONSET(P),(EONSET(P)-1)
  PP=(PPP-IONSET(P))+1
  INVOL=PNEUMO(PPP)-BASEFLOW
  RINVOL=FLOAT(INVOL)
  RINVOL=RINVOL/VOLAVE*(PBAR+PREG)/(PBAR)*310/
  (273+TA)*(PBAR+PLC-PH2O)/(PBAR+PLC-64.056)
  IF (RINVOL.LT.(0.0)) GOTO 6100
  RINVOL=0.0-RINVOL

```

```

6100      RVOLIN(PP) = RVOLIN(PP-1) - RINVOL
          RINFLOW(PP) = FLOAT(INVOL)
          RINFLOW(PP) = RINFLOW(PP) / VOLAVE * 50.0 * (PBAR + PREG) / (PBAR)
          * 310 / (273 + TA) * (PBAR + PLC-PH2O) / (PBAR + PLC-64.056)
          WRITE(*,631) PPP, RVOLIN(PP), RINFLOW(PP)
          WRITE(4,631) PPP, RVOLIN(PP), RINFLOW(PP)
610      CONTINUE
          TOPVOL = RVOLIN(PP)
          WRITE(*,630)

c
c Expiratory data analysis.
c
      DO 620 PPP = STARTEXP(P), (ENDEXP(P) - 1)
          PP = (PPP - STARTEXP(P)) + 1
          EVOL = PNEUMO(PPP) - BASEFLOW
          REXVOL = FLOAT(EXVOL)
          REXVOL = REXVOL / VOLAVE * (PBAR + PREG) / (PBAR) * 310 /
          * (273 + TA2) * (PBAR + PLC-PH2O2) / (PBAR + PLC-64.056)
          RVOLEX(PP) = RVOLEX(PP-1) + REXVOL
          VOLEX(PP) = TOPVOL - RVOLEX(PP)
          REXFLOW(PP) = FLOAT(EXVOL)
          REXFLOW(PP) = REXFLOW(PP) / VOLAVE * 50.0 * (PBAR + PREG) / (PBAR)
          * 310 / (273 + TA2) * (PBAR + PLC-PH2O2) / (PBAR + PLC-64.056)
          WRITE(*,631) PPP, VOLEX(PP), REXFLOW(PP)
          WRITE(4,631) PPP, VOLEX(PP), REXFLOW(PP)
620      CONTINUE
6      CONTINUE

c
63      FORMAT(' INSPIRATORY DATA' /
          * ' # Vol(BTPS) Flow ')
630     FORMAT('/ EXPIRATORY DATA' /
          * ' # Vol(BTPS) Flow ')
631     FORMAT(150(I4,1X,F8.4,2X,F9.4))
          CLOSE(4)

c
c Subroutines for computing FEV1.0 from the data obtained
c during the maximal flow-volume loop trials.
c
          EVOL = 0
          FEV1 = 0.0
          EXVOLINSP = 0.0
          DO 64 P = STARTEXP(1), (STARTEXP(1) + 49)
              EVOL = PNEUMO(P) - BASEFLOW
              ERVOL = FLOAT(EVOL)
              EXVOLINSP = EXVOLINSP + ERVOL
64      CONTINUE
c

```


APPENDIX 7

Example of a composite data analysis program.

Many of the comments necessary to understand the following program and its variables have been omitted, since they have been incorporated in the Appendices above.

[illegible]

C PROGRAM TO ANALYSE DATA FROM RESPIRATORY VOLUME EXPERIMENTS.

c Program uses data obtained from A/D program SPIRO-D.EXE
c using the DASH16 A/D board (MetraByte); (50 Hz, 2 channels)

```

c      INPUT DATA REQUIRED: Ch2=Trigger, Ch3=flow
c      Flow signal is both inspiratory and expiratory

```

c NOTE: THIS Program HAS OUTPUT IN S.I. UNITS
c THE ANALYSIS RELIES UPON USING A HEATED PNEUMO @ 37oC

THE FOLLOWING ANALYSES ARE PERFORMED:
(1) Tidal volumes, (2) Respiratory timing (Ti, Te, Ttot, Vt/Ti, Ti/Ttot, fb, Ve, and pauses), (3) Maximal Voluntary Ventilation, (4) Flow-Volume loops and FEV1, (5) Spirometry for determining lung volumes and capacities.

c NIGEL A.S. TAYLOR. MAY 1989
c Diving Life Support Equipment Program
c Naval Medical Research Institute
c Bethesda, MD.

c LATEST UPDATE: 4/23/90

c This program allows for unequal inspired/expired volumes
c by using only complete cycles, starting with an inspiration.
c Inspiration is detected from a negative signal from the flow
c channel.

c Program is able to analyse data obtained when subjects are immersed
c and air is provided at various hydrostatic pressures. This requires
c modification after completion of the calibration analysis routines.

[illegible]**SSORAGE:2**

INTEGER ZEROFLOW,LITER,BASELIM,PHASE1,PHASE2,INVOL
 INTEGER PNEUMO,NOFLOW,FLOW,BASE,IDATA,COMP,EXVOL
 INTEGER GAIN,PREG,PSTERN,IONSET,EONSET,TIME,MAX
 INTEGER MAX2,IVOL,BASEFLOW,PRESS,INFLOW,EXFLOW
 INTEGER A,B,C,D,E,F,I,J,K,L,M,N,P,PP,PPP,Q,R,S,T,W,X,Y,Z


```

WRITE(*,7004)
7004 FORMAT(' ENTER LOW GAIN BASAL CALIBRATION FILENAME: '/')
READ(*,7005) NAME1
7005 FORMAT(A40)
WRITE(*,7006)
7006 FORMAT(' ENTER LOW GAIN FLOW CALIBRATION FILENAME: '/')
READ(*,7005) NAME2
WRITE(*,7007)
7007 FORMAT(' ENTER HIGH GAIN BASAL CALIBRATION FILEMANE: '/')
READ(*,7005) NAME3
WRITE(*,7008)
7008 FORMAT(' ENTER HIGH GAIN FLOW CALIBRATION FILENAME: '/')
READ(*,7005) NAME4
7002 CONTINUE
1310 WRITE(*,1)
1 FORMAT(///
* ' WHICH TYPE OF DATA DO YOU WISH TO ANALYSE?'/
* ' 1 RETURN = SPONTANEOUS RESPIRATION'/
* ' 3 RETURN = LUNG VOLUME SPIROMETRY'/
* ' 6 RETURN = MAXIMAL VOLUNTARY VENTILATION'/
* ' 9 RETURN = MAXIMAL FLOW-VOLUME LOOPS'/)
READ(*,100) A

```

C

c Determining data file dimensions and setting basal input
c according to the gain settings used for different experiments

C

```

IF (A.NE.1.AND.A.NE.3) GOTO 1991
IF (A.EQ.1) TIME=60*50*2
IF (A.EQ.3) TIME=45*50*2
FILEBASE=NAME1
FILEFLOW=NAME2
FLOW=-300
ARBITLIM=1500
GOTO 1990
1991 IF (A.EQ.6) TIME=15*50*2
IF (A.EQ.9) TIME=12*50*2
FILEBASE=NAME3
FILEFLOW=NAME4
FLOW=-500
ARBITLIM=500
1990 MAX2=TIME
MAX=MAX2/2

```

C

```

SYRINGE = 3.000
OPEN(3,FILE=FILEBASE,STATUS='OLD')
READ(3,101) ZEROFLOW
WRITE(*,102) ZEROFLOW
101 FORMAT(I7)
102 FORMAT(' BASAL FLOW (RAW UNITS) = ',I5//)
CLOSE(3)
50 FORMAT(2X,F9.2,2X,F9.2)

```



```

C
2   FORMAT(// 'PLEASE NAME THE DATA FILE TO BE ANALYSED.'//)
   *   '   examples: (1) BARNES.RES = SPONTANEOUS BREATHING'/
   *   '   (3) BARNES.SPI = LUNG VOLUME SPIROMETRY'/
   *   '   (6) BARNES.MVV = MAXIMAL VOLUNTARY VENTILATION'/
   *   '   (9) BARNES.FLO = MAXIMAL FLOW-VOLUME LOOPS'//)
20  FORMAT(A40)
c
   WRITE(*,130)
130  FORMAT(' WHAT WAS THE BAROMETRIC PRESSURE (mmHg F8.4)?'/)
   READ(*,131) PBAR
131  FORMAT(F12.3)
   PBAR = PBAR * 1.36
c
c The question below is used only if inspired air temp was not 37 °C
c TA = inspired gas temperature, TA2 = expired temp, PH2O = water
c vapour pressure, PH2O2 = expired water vapour pressure. Both pressures
c are converted to cmH2O.
c
C   WRITE(*,132)
C132 FORMAT(' WHAT WAS THE INSPIRED AIR TEMPERATURE (deg F F8.4)?'/)
C   READ(*,131) TA
   TA = 37.0
   TA2 = 37.0
   PH2O2 = 47.1 * 1.36
   WRITE(*,133)
133  FORMAT(' WHAT WAS INSPIRED WATER VAPOR PRESSURE (mmHg F8.4)?'/)
   READ(*,131) PH2O
   PH2O = PH2O * 1.36
c
c Variables used when air is provided at various breathing pressures.
c COMP is compensator pressure used to balance the differential transducer.
c This is necessary when mouth pressures during immersion exceed the range
c limits of the transducer. PLC = lung centroid pressure, COMPRESS and
c SIPLC are SI equivalents of the above pressures. PREG = hydrostatic
c pressure at center of the regulator (ie. depth in cmH2O), PSTERN =
c hydrostatic pressure at the depth of the sternal notch, PHASE = the
c phase shift between flow and pressure signals (not used).
c
   COMP = 0
   COMPRESS = 0.0
   PLC = 0
   SIPLC = 0
   PREG = 0
   PSTERN = 0
   PHASE = 0
c
1311 WRITE(*,2)
     READ(*,20) FILE1
c
c The output file is in ASCII format, and with the extension .PRN

```



```

        VOLBTPS(J)=0.0
        IONSET(J)=0
        EONSET(J)=0
        TINSP(J)=0.0
        TEXP(J)=0.0
        TTOTAL(J)=0.0
3002    CONTINUE
c
        TOTALVOL = 0.0
c
        I=0
        E=0
        M=1
        L=0
c
3      DO 3111 J=M,MAX
        IF (PNEUMO(J).GE.BASELIM) GOTO 3112
3111    CONTINUE
3112    DO 31 K=J,MAX
        IF (PNEUMO(K).LT.BASELIM) GOTO 310
31      CONTINUE
c
310     IF (K.GE.MAX) GOTO 3102
c
C       WRITE(*,5500) K,PNEUMO(K)
C5500   FORMAT(' IONSET AT POINT 'I5,' = 'I5)
C
        VOLINSP=0.0
        VOL2INSP=0.0
        EXVOLINSP=0.0
        EX2VOLINSP=0.0
c

        DO 32 T=K,MAX
        IF (PNEUMO(T).GE.BASELIM) GOTO 33
        IVOL=PNEUMO(T)-BASEFLOW
        RVOL=FLOAT(IVOL)
        VOLINSP = VOLINSP - RVOL
32      CONTINUE
33      CONTINUE
c
C       WRITE(*,5501) VOLINSP
C5501   FORMAT(' INTEGRATED TOTAL = ',F10.2)
c
        IF (VOLINSP.LT.ARBITLIM) GOTO 3103
c
        I=I+1
        IONSET(I)=K
        EONSET(I)=T
        DO 330 J=1,15
        VOL(J)=0.0

```

```

330    CONTINUE
c
      L=L+1
      VOL(L) = VOLINSP / VOLAVE
c
c    Volume correction for regulator pressure and PLC, then BTPS correction
c    (all values are in cmH2O and degC units).
c
      VOL(L) = VOL(L) * (PBAR+PREG) / (PBAR)
      VOLBTPS(L)=VOL(L)*310/(273+TA)*(PBAR+PLC-PH2O)
      •      /(PBAR+PLC-64.056)
      TIDALV(L)=VOLBTPS(L)
c
3103  IF (T.GE.MAX) GOTO 3102
      IF (VOLINSP.GT.ARBITLIM) GOTO 3104
      M = T
      GOTO 3
c
3104  DO 34 EX=T,MAX
      IF (PNEUMO(EX).GT.TBASELIM) GOTO 340
34    CONTINUE
340   CONTINUE
c
      DO 341 EXP=EX,MAX
      IF (PNEUMO(EXP).LE.TBASELIM) GOTO 3410
      EVOL=PNEUMO(EXP)-BASEFLOW
      ERVOL=FLOAT(EVOL)
      EXVOLISNP = EXVOLINSP + ERVOL
341   CONTINUE
3410  CONTINUE
c
      IF (EXP.GE.MAX) GOTO 3102
      IF (EXVOLINSP.GT.ARBITLIM) GOTO 3411
      T = EXP
      GOTO 3104
c
3411  E=E+1
      STARTEXP(E)=EX
      ENDEXP(E)=EXP
c
      EXTIDALV(L)=EXVOLINSP/VOLAVE*(PBAR+PREG)/(PBAR)*310/(273+TA2)
      •      *(PBAR+PLC-PH2O2)/(PBAR+PLC-64.056)
c
      M=T
      GOTO 3
3102  CONTINUE
c
      IF (A.EQ.9) GOTO 6000
c
c    Maximal voluntary ventilation calculations
c

```



```

WRITE(4,420)
WRITE(*,420)
420  FORMAT(' TIDAL/I TIDAL/E INSP/Time EXP/Time T/tot '
      * ' POST-INSP/Pause POST-EXP/Pause')
c
DO 421 P=1,(I-1)
WRITE(*,422) TIDALV(P),EXTIDALV(P),TINSP(P),TEXP(P),TTOTAL(P),
      * POSTINSP(P),POSTEXP(P)
WRITE(4,422) TIDALV(P),EXTIDALV(P),TINSP(P),TEXP(P),TTOTAL(P),
      * POSTINSP(P),POSTEXP(P)
421  CONTINUE
422  FORMAT(F7.3,2X,F7.3,2X,F7.4,1X,F7.4,1X,F7.4,5X,F7.4,8X,F7.4)
c
AVTINSP=0.0
AVTEXP=0.0
AVTTOTAL=0.0
AVTIDALV = 0.0
AVINSPPOST = 0.0
AVEXPPOST = 0.0
DO 44 P=1,(I-1)
AVTIDALV = AVTIDALV + TIDALV(P)
AVTINSP = AVTINSP + TINSP(P)
AVTEXP = AVTEXP + TEXP(P)
AVTTOTAL = AVTTOTAL + TTOTAL(P)
AVINSPPOST = AVINSPPOST + POSTINSP(P)
AVEXPPOST = AVEXPPOST + POSTEXP(P)
44  CONTINUE
c
AVTIDALV = AVTIDALV / (I-1)
AVTINSP = AVTINSP / (I-1)
AVTEXP = AVTEXP / (I-1)
AVTTOTAL = AVTTOTAL / (I-1)
RFREQ=60.0/AVTTOTAL
VTIMEINSP = AVTIDALV / AVTINSP
MINVENT = AVTIDALV * RFREQ
TINTOTAL = AVTINSP / AVTTOTAL
AVINSPPOST = AVINSPPOST / (I-1)
AVEXPPOST = AVEXPPOST / (I-1)
c  MINV2 = VTIMEINSP * TINTOTAL
c
WRITE(*,440) AVTIDALV
WRITE(4,440) AVTIDALV
440  FORMAT('AVERAGE TIDAL VOLUME = ',F12.4/)
WRITE(*,441) AVTINSP
WRITE(4,441) AVTINSP
441  FORMAT('AVERAGE TIME INSP = ',F12.4)
WRITE(*,442) AVTEXP
WRITE(4,442) AVTEXP
442  FORMAT('AVERAGE TIME EXPIR. = ',F12.4)
WRITE(*,443) AVTTOTAL
WRITE(4,443) AVTTOTAL
443  FORMAT('AVERAGE TOTAL TIME = ',F12.4/)

```



```

6100    RVOLIN(PP) = RVOLIN(PP-1) - RINVOL
        RINFLOW(PP) = FLOAT(INVOL)
        RINFLOW(PP) = RINFLOW(PP) / VOLAVE * 50.0 * (PBAR + PREG) / (PBAR)
        * 310 / (273 + TA) * (PBAR + PLC-PH2O) / (PBAR + PLC-64.056)
        WRITE(*,631) PPP, RVOLIN(PP), RINFLOW(PP)
        WRITE(4,631) PPP, RVOLIN(PP), RINFLOW(PP)
610    CONTINUE
c
        TOPVOL = RVOLIN(PP)
c
        WRITE(*,630)
c        WRITE(4,630)
c
        DO 620 PPP = STARTEXP(P), (ENDEXP(P) - 1)
            PP = (PPP - STARTEXP(P)) + 1
            EXVOL = PNEUMO(PPP) - BASEFLOW
            REXVOL = FLOAT(EXVOL)
            REXVOL = REXVOL / VOLAVE * (PBAR + PREG) / (PBAR) * 310 /
            * (273 + TA2) * (PBAR + PLC-PH2O2) / (PBAR + PLC-64.056)
            RVOLEX(PP) = RVOLEX(PP-1) + REXVOL
            VOLEX(PP) = TOPVOL - RVOLEX(PP)
            REXFLOW(PP) = FLOAT(EXVOL)
            REXFLOW(PP) = REXFLOW(PP) / VOLAVE * 50.0 * (PBAR + PREG) / (PBAR)
            * 310 / (273 + TA2) * (PBAR + PLC-PH2O2) / (PBAR + PLC-64.056)
            WRITE(*,631) PPP, VOLEX(PP), REXFLOW(PP)
            WRITE(4,631) PPP, VOLEX(PP), REXFLOW(PP)
620    CONTINUE
c
6    CONTINUE
c
63    FORMAT(' INSPIRATORY DATA'/
        * ' # Vol(BTPS) Flow ')
630    FORMAT('/ EXPIRATORY DATA'/
        * ' # Vol(BTPS) Flow ')
631    FORMAT(150(14,1X,F8.4,2X,F9.4))
        CLOSE(4)
c
c Subroutines for computing FEV1.0.
c
        EVOL = 0
        FEV1 = 0.0
        EXVOLINSP = 0.0
C
        DO 64 P = STARTEXP(1), (STARTEXP(1) + 49)
            EVOL = PNEUMO(P) - BASEFLOW
            ERVOL = FLOAT(EVOL)
            EXVOLINSP = EXVOLINSP + ERVOL
64    CONTINUE
c
        FEV1 = EXVOLINSP / VOLAVE * (PBAR + PREG) / (PBAR) * 310
        * / (273 + TA2) * (PBAR + PLC-PH2O2) / (PBAR + PLC-64.056)
c

```


APPENDIX 8

Example analysis program for use during an exercise protocol.

Many of the comments necessary to understand this program and its variables have been omitted where they have been provided in preceeding Appendices.

```

c
c
c  Program TO ANALYSE RESPIRATORY VOLUME AND MOUTH PRESSURE DATA.
c
c  Program uses data obtained from A/D program
c  using the DASH16 A/D board (MetraByte, at 50Hz),
c  and produces breath-by-breath and 20 sec average output.
c
c  ASCII INPUT DATA REQUIRED:
c  ASCII file contains 2 columns of data:
c  Column 1 = mouth pressure, Column 2 = flow.
c
c  THE FOLLOWING ANALYSES ARE PERFORMED:
c  (1) Tidal volumes, (2) Respiratory timing (Ti, Te, Ttot,
c  Vt/Ti, Ti/Ttot, fb), (3) Minute ventilation, (4) Peak-to-peak
c  mouth pressures (cmH2O), and (5) flow-volume-pressure data
c  for the first breath of each 20 sec interval.
c
c  Limitations: (1) Expiratory data cannot be measured using this
c  program without changes. It has been designed
c  for a set-up using inspiratory pneumotach only.
c  (2) Data is read in from an ASCII file input only.
c  (3) Convention for flow: inspiration is negative.
c
c  NIGEL A.S. TAYLOR. June 1990
c  Diving Medicine, Naval Medical Research Institute
c
c  LATEST UPDATE: 10/12/90
c
c  This program allows for unequal inspired/expired volumes
c  by using only complete cycles, starting with an inspiration.
c  Inspiration is detected from a negative signal from the flow
c  channel. The phase shift between the flow and the pressure
c  signals is taken into account during analysis.
c
c  Three types of data output files are used:
c  (1) file holding breath-by-breath data (variables 1 to 4);
c  (2) file holding flow-volume-pressure data (20 msec intervals);
c  (3) a database holding only mean values from variables in file 1.
c  These correspond with Units 7, 6 and 3 respectively in program.
c
c
c
c  $STORAGE:2

```



```

WRITE(*,1002)
1002  FORMAT(/' ENTER NAME OF PRESSURE CALIBRATION FILE: '/')
      READ(*,20) FILE4
C
C   Reading calibration data from files.
C
      OPEN (6,FILE = FILE4,STATUS='OLD')
      READ(6,50) REGCOEFF,CONSTANT
      CLOSE(6)
      WRITE(*,5099) REGCOEFF,CONSTANT
5099  *  FORMAT(/ REGRESSION COEFFICIENT (cmH2O) = 'F9.2/
      '   CONSTANT A (cmH2O) = 'F9.2//)
C
      OPEN(3,FILE = FILE1,STATUS='OLD')
      READ(3,101) ZEROFLOW
      CLOSE(3)
C
      WRITE(*,102) ZEROFLOW
101   FORMAT(I7)
102   FORMAT(' BASAL FLOW (RAW UNITS) = 'I5//)
50    FORMAT(2X,F9.2,2X,F9.2)
C
      BASE = (FLOW - ZEROFLOW) * 4 / 100
      BASELIM = BASE + ZEROFLOW
      TBASELIM = ZEROFLOW - BASE
C
      WRITE(*,103) BASE
103   FORMAT(' VALUE OF BASE = 'I5//)
C
      WRITE(*,104) BASELIM
104   FORMAT(' VALUE OF BASELIMIT = 'I5//)
C
      OPEN(3,FILE = FILE2,STATUS='OLD',FORM='UNFORMATTED')
      READ(3) (LITER(J),J = 1,2000)
11    FORMAT(10(I7))
      CLOSE(3)
C
      M = 1
      L = 0
12    DO 120 J = M,2000
      IF (LITER(J).LT.BASELIM) GOTO 1200
120   CONTINUE
1200  CONTINUE
      IF (J.GE.2000) GOTO 129
C
C   Integration of flow data.
      VOLINSP = 0.0
121   DO 127 T = J,2000
      IVOL = LITER(T) - ZEROFLOW
      RVOL = FLOAT(IVOL)
      VOLINSP = VOLINSP + RVOL
      IF (LITER(T).GE.BASELIM) GOTO 1270
127   CONTINUE
1270  CONTINUE
C
C   Checking data for false volumes before storing volumes.

```

```

IF (VOLINSP.LE.ARBITLIM) GOTO 128
L=L+1
CALVOL(L)=VOLINSP
C WRITE(*,1391) L,CALVOL(L)
1391 FORMAT(' FOR PUMP ',I3,' CALVOL = ',F12.1)
128 IF (T.GE.2000) GOTO 129
M=T
GOTO 12

c                                     Calculation of average calibration value.
129 RVOLSUM=0.0
DO 1280 W=1,L
RVOLSUM=RVOLSUM+CALVOL(W)
1280 CONTINUE
VOLAVE = RVOLSUM / (SYRINGE*L)
1281 FORMAT(10(I7))
WRITE(*,1282) VOLAVE
1282 FORMAT('/ MEAN FLOW CALIBRATION VALUE = ',F14.4,' bits/litre'//)
WRITE(*,1360)
1360 FORMAT('// LOOK AT THE VOLUME CALIBRATION VALUE:/'
* ' SELECT A FALSE VOLUME LIMIT BASED UPON THIS VALUE:/'
* ' [e.g. 200 ml = 20%]'//
* ' ENTER YOUR VALUE {I7 - NO decimal}'//)
READ(*,1361) LIMITVOL
1361 FORMAT(I9)
C
2 FORMAT('// ENTER NAME OF DATA FILE TO BE ANALYSED:////)
20 FORMAT(A10)
14 FORMAT(/// HOW MANY MINUTES DID THIS EXPERIMENT RUN? [I3]'//)
WRITE(*,130)
130 FORMAT('/ WHAT WAS THE BAROMETRIC PRESSURE (mmHg F12.2)?'//)
READ(*,131) PBAR
131 FORMAT(F12.2)
PBAR = PBAR * 1.36
WRITE(*,132)
132 FORMAT('// WHAT WAS THE INSPIRED AIR TEMPERATURE (degC F8.4)?'//)
READ(*,131) TA
WRITE(*,133)
133 FORMAT('// WHAT WAS INSPIRED WATER VAPOUR PRESSURE (mmHg F8.4)?'//)
READ(*,131) PH2O
PH2O = PH2O * 1.36

c

COMP = 0
COMPRESS = 0.0
PLC = 0
SIPLC = 0
PREG = 0
PSTERN = 0
PHASE1 = 0

c PHASE1 comment:
c The flow and mouth pressure signals are rarely in phase so
c this difference must be taken into account. Let the value for

```

C

✱

C

C

C

C

```

c      ETIME = ETIME + 0.3334
      TIME = TIME + 1
      POINT = 1
      POINT2 = 1000
      BASEFLOW = ZEROFLOW
      J=0
      K=0

c
      DO 21 K=POINT,POINT2
      READ(3,2090,END=2999) (IDATA(L,K),L=1,2)
21      CONTINUE
      GOTO 2998
2090      FORMAT(2I6)
2999      WRITE(*,24) FILENAME
2998      CONTINUE
C
      DO 22 J=POINT,POINT2
      PRESS(J)=IDATA(1,J)
      PNEUMO(J)=IDATA(2,J)
22      CONTINUE
C
C      WRITE(*,2292) IDATA(1,POINT)
2292      FORMAT(' STARTING POINT FOR 20 SEC ',I5)
C      WRITE(*,2293) J
2293      FORMAT(' FINISHING POINT FOR 20 SEC ',I4)
C
      GOTO 6003

C
24      FORMAT('                END OF FILE ENCOUNTERED:/'
*          ' ANALYSIS OF FILE: ',A10,' COMPLETE.'//)

c
c      Volume analysis routine.
6003      N=0

c
      DO 3002 J=1,15
      TIDALV(J)=0.0
      VOLBTPS(J)=0.0
      IONSET(J)=0
      EONSET(J)=0
      TINSP(J)=0.0
      TEXP(J)=0.0
      TTOTAL(J)=0.0
3002      CONTINUE
      TOTALVOL = 0.0

c
c      COUNTERS: I = inspiration onset, E = expiration onset,
c      P = pairs of insp and exp, J,K,M = starting points,
c      L = tidal volumes.
c      NOTE: I is used as a respiratory cycle counter from this point.

```

c Integrating flow to obtain volume (inspiration -ive).

c

E=0

I=0

J=0

K=0

L=0

M=1

C

3

CONTINUE

C

DO 3111 J=M,POINT2

IF (PNEUMO(J).GE.BASELIM) GOTO 3112

WRITE(*,8820) PNEUMO(J),J

C

3111

CONTINUE

88220

FORMAT(' + PNEUMO(J) = ',I6,' FOR J = ',I6)

C

WRITE(*,8822)

8822

FORMAT('// ' FINISHED LOOP '//)

C

3112

DO 31 K=J,POINT2

IF (PNEUMO(K).LT.BASELIM) GOTO 310

C

WRITE(*,8820) PNEUMO(J),J

31

CONTINUE

C

310

CONTINUE

C

WRITE(*,8822)

IF (K.GE.POINT2) GOTO 3102

VOLINSP=0.0

EXVOLINSP=0.0

c

DO 32 T=K,POINT2

IF (PNEUMO(T).GE.BASELIM) GOTO 33

IVOL=PNEUMO(T)-BASEFLOW

RVOL=FLOAT(IVOL)

VOLINSP = VOLINSP - RVOL

32

CONTINUE

33

CONTINUE

c

The -RVOL converts volume to positive.

c

Check for small, false volumes.

c

IF (VOLINSP.LT.ARBITLIM) GOTO 3103

c

c Marking points of inspiration start (IONSET), & end (EONSET),

c and expiration start (STARTEXP), & end (ENDEXP).

c

I=I+1

IONSET(I)=K

EONSET(I)=T


```

c      TOPVOL=0.0
      BIT = 0.0
      EBIT = 0.0
c
c      DO 500 P=1,ZZZ
c
c          IF (P.NE.1) GOTO 500
c
c              DO 501 J=1,250
c                  RVOLIN(J)=0.0
c                  RINFLOW(J)=0.0
501      CONTINUE
c                  DO 5011 J=1,900
c                      MPRESS(J)=0.0
c                      PPEAK(J)=0.0
c                      EMPRESS(J)=0.0
c                      EPEAK(J)=0.0
5011     CONTINUE
c
c
DO 510 PPP=IONSET(P),(EONSET(P)-1)
    PP=(PPP-IONSET(P))+1
    BIT = BIT + 0.02
    IPRESS=PRESS(PPP+PHASE1)
    MPRESS(PP)=FLOAT(IPRESS)
    MPRESS(PP)=(MPRESS(PP)-CONSTANT)/REGCOEFF
    INVOL=PNEUMO(PPP)-BASEFLOW
    RINVOL=FLOAT(INVOL)
    RINVOL=RINVOL/VOLAVE*(PBAR + PREG)/(PBAR)*310/
    * (273+TA)*(PBAR + PLC-PH2O)/(PBAR + PLC-64.056)
    IF (RINVOL.LT.(0.0)) GOTO 5100
    RINVOL=0.0-RINVOL
5100   RVOLIN(PP)=RVOLIN(PP-1)-RINVOL
    RINFLOW(PP)=FLOAT(INVOL)
    RINFLOW(PP)=RINFLOW(PP)/VOLAVE*50.0*(PBAR + PREG)/(PBAR)
    * 310/(273+TA)*(PBAR + PLC-PH2O)/(PBAR + PLC-64.056)
    IF (TIME.EQ.1) GOTO 5110
    IF (TIME.EQ.5) GOTO 5110
    IF (TIME.EQ.11) GOTO 5110
    IF (TIME.EQ.17) GOTO 5110
    IF (TIME.EQ.23) GOTO 5110
    IF (TIME.EQ.29) GOTO 5110
    IF (TIME.EQ.35) GOTO 5110
    IF (TIME.EQ.41) GOTO 5110
    IF (TIME.EQ.47) GOTO 5110
    IF (TIME.EQ.53) GOTO 5110
    IF (TIME.EQ.59) GOTO 5110

```



```

        DO 6111 J=1,15
            PPEAK(J)=0.0
            RATIO(J)=0.0
6111    CONTINUE
c
        DO 6 P=1,ZZZ
c
            DO 600 PPP=IONSET(P),(EONSET(P)-1)
                PP=(PPP-IONSET(P))+1
                IPRESS=PRESS(PPP+PHASE1)
                MPRESS(PP)=FLOAT(IPRESS)
                MPRESS(PP)=(MPRESS(PP)-CONSTANT)/REGCOEFF
                IF (MPRESS(PP).GE.PEAK(COUNT)) GOTO 6000
                COUNT = COUNT + 1
                PEAK(COUNT) = MPRESS(PP)
                WRITE(*,6401) PEAK(COUNT)
c
6000    CONTINUE
600
c
c
c        IF (PEAK(COUNT).LT.(-100.0)) PEAK(COUNT)=0.0
c        WRITE(*,6400) PEAK(COUNT)
6400    FORMAT('      PEAK INSPIRATORY PRESSURE = ',F9.3)
6404    FORMAT('      PEAK EXPIRATORY PRESSURE = ',F9.3)
6401    FORMAT('+ INSPIRATORY PRESSURE REAL =',F9.1)
6403    FORMAT('+ EXPIRATORY PRESSURE REAL =',F9.1)
6405    FORMAT(' PEAK-TO-PEAK= 'F9.1' MEAN='F9.1' BREATH 'I3/)
c
        DO 610 PPP=EONSET(P),(IONSET(P+1)-1)
            PP=(PPP-EONSET(P))+1
            EPRESS=PRESS(PPP+PHASE1)
            EMPRESS(PP)=FLOAT(EPRESS)
            EMPRESS(PP)=(EMPRESS(PP)-CONSTANT)/REGCOEFF
            IF (EMPRESS(PP).LE.EPEAK(ECOUNT)) GOTO 6100
            ECOUNT = ECOUNT + 1
            EPEAK(ECOUNT) = EMPRESS(PP)
            WRITE(*,6403) EPEAK(ECOUNT)
c
6100    CONTINUE
610
c
c
c        IF (EPEAK(ECOUNT).GT.(100.0)) EPEAK(ECOUNT)=0.0
c        WRITE(*,6404) EPEAK(ECOUNT)
c
        PPEAK(P)=EPEAK(ECOUNT)-PEAK(COUNT)
c        WRITE(*,6402) PPEAK(P),P
6402    FORMAT(' PEAK-TO-PEAK PRESSURE = ',F9.3, ' BREATH'I4)
        IF (EPEAK(ECOUNT).LE.(0.0)) GOTO 6138
        RATIO(P)=(0.0-PEAK(COUNT))/EPEAK(ECOUNT)
        GOTO 6139
6138    RATIO(P)=0.0
6139    CONTINUE
c
        WRITE(7,41) P,IONSET(P),EONSET(P),TIDALV(P),PPEAK(P)

```



```
IF (A.EQ.9) GOTO 9999
IF (A.EQ.1) GOTO 1310
IF (A.EQ.6) GOTO 1006
9999 STOP
END
```

GLOSSARY OF TERMS AND ABBREVIATIONS

Lung volume and ventilation variables (liters BTPS):

ERV	= expiratory reserve volume
FRC	= functional residual capacity
IC	= inspiratory capacity
IRV	= inspiratory reserve volume
RV	= residual volume
TLC	= total lung capacity
VC	= vital capacity
V_T	= tidal volume
V_E	= minute ventilation derived from expiratory data
MVV	= maximal voluntary ventilation
$FEV_{1.0}$	= forced expired volume during first second of maximal expiration
$FEV_{1.0}/VC$	= ratio of $FEV_{1.0}$:vital capacity expressed as a percentage.

Respiratory flow variables (liters/second⁻¹):

PEFR	= peak expiratory flow rate
$MEFR_{25}$	= maximum expiratory flowrate at 25% of VC
$MEFR_{50}$	= maximum expiratory flowrate at 50% of VC
$MEFR_{75}$	= maximum expiratory flowrate at 75% of VC.

Respiratory timing variables (seconds):

T_I	= time of inspiration
T_E	= time of expiration
T_{TOT}	= total respiratory duration

Post-exp pause = duration of the pause between the end of expiration and the commencement of the next inspiration

Post-insp pause = duration of the pause between the end of inspiration and the commencement of the next expiration

T_I/T_{TOT} = ratio of the time of inspiration to the total cycle duration

V_T/T_I = ratio of the tidal volume to the inspiratory duration ($l.s^{-1}$).