NOAA Technical Memorandum ERL WPL-170

DEPARTMENT OF COMMERCE
UNITED STATES OF AMERICA

DATA ACQUISITION AND ANALYSIS FOR THE 1988 MICROMETEOROLOGICAL SCINTILLATION EXPERIMENT

J. Tant Priestley

Wave Propagation Laboratory
Boulder, Colorado

August 1989

noaa NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION / Environmental Research Laboratories

QC
807.5
.U6
W6
no. 170

NOAA Technical Memorandum ERL WPL-170

DATA ACQUISITION AND ANALYSIS FOR THE 1988 MICROMETEOROLOGICAL SCINTILLATION EXPERIMENT

J. Tant Priestley

Wave Propagation Laboratory
Boulder, Colorado
August 1989

UNITED STATES
DEPARTMENT OF COMMERCE

Robert A. Mosbacher
Secretary

NATIONAL OCEANIC AND
ATMOSPHERIC ADMINISTRATION

Environmental Research
Laboratories

Joseph O. Fletcher
Director

CONTENTS

DATA ACQUISITION AND ANALYSIS FOR THE 1988 MICROMETEOROLOGICAL SCINTILLATION EXPERIMENT

J. Tant Priestley*

ABSTRACT

The purpose of the 1988 Micrometeorological Scintillation Experiment was to assess the ability of several scintillation techniques to measure the heat and momentum fluxes and the stability of the atmospheric surface layer. This report documents the data acquisition and analysis software used for the experiment. An overview of the software modules is given, including their purposes and relationships to one another. A more detailed description of each module is also given. Selected portions of the software source code are provided in the appendices, and the complete source code is provided on an enclosed diskette.

1. INTRODUCTION

The surface fluxes of heat, moisture, and momentum are the basic way in which the atmosphere interacts with the Earth's surface. This interaction is a driving force behind the Earth's weather and climate. Remote sensing of these fluxes using the scintillation of electromagnetic waves promises to be a superior means of monitoring them. Remote sensing is superior because it gives an area average, rather than a point measurement, it is not affected by contamination as are in situ probes, and it is unaffected by flow distortion around the mounting platform of the remote sensing instrument. These two latter effects are particularly severe in over-ocean measurements because of salt-spray contamination, and because only large mounting platforms (e.g., ships or oil-drilling platforms) are usually available.

The 1988 Micrometeorological Scintillation Experiment took place at Table Mountain, about 10 km north of Boulder, Colorado. The purpose of the experiment was to assess the ability of three novel scintillation techniques to measure surface-layer stability and the fluxes of heat and momentum. The connection between scintillation measurements and these fluxes and stability depends on application of the Monin-Obukhov similarity theory of the surface layer. One scintillation technique used a measurement of laser scintillation observed through a small receiver aperture combined with a similar measurement of scintillation using a large-aperture incoherent-radiation scintillometer operating on the same propagation path. This combined system constitutes an optical inner-scale meter. It produces measurements of both turbulence inner scale and refractive structure parameter, C_n^2; these quantities are then used to determine the heat and momentum fluxes and stability. The second technique used two large-aperture C_n^2 scintillometers operating at different heights; the stability can be deduced from the ratios of the two values of C_n^2. The third technique used an optical cross-wind scintillometer having its measurement axis turned vertically so that the scintillation-derived average vertical component of the wind was determined; we investigate whether or not the stability can be determined from this

* Present affiliation NSR, Boulder, Colorado.

measurement. In situ instrumentation was deployed for comparison with results from the scintillation techniques. This in situ instrumentation included a three-axis sonic anemometer; a propeller-vane anemometer; platinum resistance wire thermometers for single-point temperature fluctuations as well as two-point temperature structure parameter determination; spaced hot-film anemometers for the velocity structure parameter; and a barometer.

The primary purpose of this report is to document the data acquisition and analysis software. Included are (1) an overview of the four major software modules, their purposes, and how they interact, (2) a more detailed description of each individual module, and (3) the actual source code of the software (selected portions in appendices and the complete software on the enclosed diskette). The theoretical basis for these calculations is covered in Appendix A.

When questions arise about some computed result, it will be possible from this report to determine exactly how that result was calculated. If it is determined that it should be computed differently, or if it is desirable to compute additional parameters, this report will provide the necessary documentation for such changes or additions. Finally, if the experiment is ever repeated—with improved instruments or under different meteorological conditions—this report will make it possible to reuse the same software to modify the software.

This report can also serve an important secondary purpose. The overall system design should be useful as a model for the software design of other field experiments. Also, many of the principles used in the design and coding of the individual modules should be helpful for a broad spectrum of programming activities. A discussion of the overall system design is given in Section 2; some of the design and programming principles that I have found to be generally useful are illustrated in Section 6. Many of these techniques are derived from the literature on "structured programming," (McCracken, 1972; Pressman, 1987) although the one I have found the most helpful, the categorization and partitioning of a subprogram's formal parameters (see Appendix B), I devised about a dozen years ago.

2. SYSTEM OVERVIEW

The purpose of the system is first to acquire data from a set of meteorological instruments and second to process these data to test the usefulness of scintillation techniques for measuring certain basic meteorological parameters.

The system overview is illustrated in the data-flow diagram in Fig. 1. This figure primarily shows the software modules, in four of the five rectangular boxes (S1–S4), and the data flow on the various connecting lines. Certain of the data-flow lines indicate file transfers and thus represent data files that can be stored on or transferred by diskettes. The other data-flow lines indicate physical electrical connections.

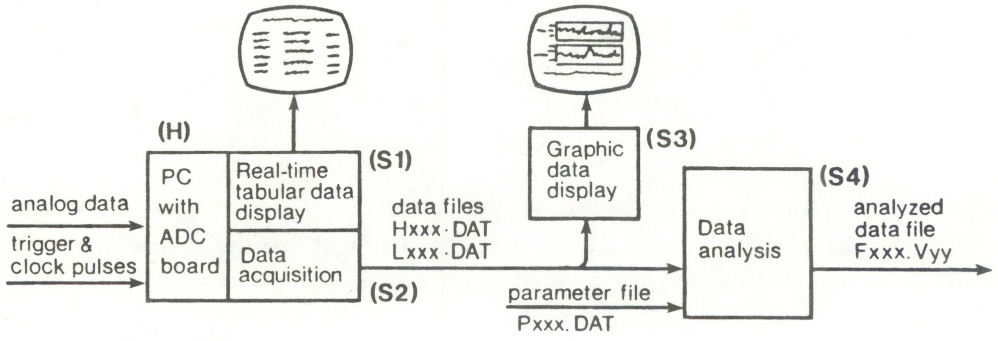

Fig. 1. Data-flow diagram of the data acquisition and processing system.

Tracing the data flow, two lines enter the hardware box H at the left of the figure. The upper line represents the 16 analog data lines from the meteorological instruments, and the lower line represents the trigger and clock signals needed by the analog-to-digital converter (ADC). (Appendix C gives more details about the trigger and clock pulses.) The hardware box labeled "PC with ADC board" is a WIN Turbo AT, an IBM-AT clone personal computer (PC), with a Data Translation series DT2801 board installed. The ADC board contains a 16–channel single-ended multiplexer followed by a 12–bit, analog-to-digital converter.

Continuing to trace the data flow in Fig. 1, the software module S2 "Data acquisition" performs the data acquisition function, creating two data files Hxxx.DAT and Lxxx.DAT for each run. The H and L in the file names indicate the High-frequency and Low-frequency data files, and the xxx indicates the three-digit run number. The high-frequency file Hxxx.DAT contains the data from channels 1–4, the fast-response instruments, and the low-frequency file Lxxx.DAT contains the data from channels 5–16, the slow-response instruments. (Channels 5–16 are sometimes referred to in this document as the **low frequency** channels 1–12.) These two data files are then stored on a 1.2 megabyte (Mbyte) diskette designated for that run.

Finally, the conversion to engineering units (see Appendix D) and data analysis are performed by software module S4 "Data analysis." Typically this analysis is performed on a faster PC (an Everex Step 386/20) back at our Boulder laboratory. The operating procedure is first to create a parameter file Pxxx.DAT (see Appendix E for an example) that contains various operational parameters for the run, then to insert a data diskette and start the program. This module analyzes the data according to the specifications given in

Appendix A of this report. The resulting output file is Fxxx.Vyy (see Appendix F) where the F in the file name stands for Final, xxx is the run number, and yy specifies the version number of the processing program FLUX1.

The remaining two software modules S1 and S3 are support modules. S1 "Real-time tabular data display" gives a real-time screen display of all of the voltages out of the instruments along with these outputs converted into engineering units. This module was extremely useful, both during the initial setup of the instruments and for checking the instruments before and several times during each daily session.

Module S3 provides the means for viewing the actual time series of any data channel. This type of capability is almost mandatory for explaining unusual meteorological events or just determining if some particular instrument was working.

3. REAL-TIME TABULAR DATA DISPLAY

The Real-Time Tabular Data Display (Fig. 1, box S1) is a support module used in the setup and monitoring of the meteorological instruments. When the program is running, a table is displayed on the computer screen that gives the names of the 16 input channels, the voltages from the channels, and those voltages converted into the appropriate engineering units. The values in the table are updated approximately once a second, so it is possible for a person to watch the output of an instrument on the screen while another person makes adjustments on that instrument. It is also convenient to view briefly the outputs of all the instruments before a data run.

3.1 Organization of Program Files

The operational program files READP.EXE, CC.EXE, and DISP_CH1.EXE and the procedure files DISP.BAT and DISP_CH.BAT are all located in directory \FLUX\EXE on the field computer. The source files CC.C, VP.H, and DTLIB.C, all of which are part of program CC, the FORTRAN source files READP.FOR and DISP_CH1.FOR, and the procedure files DISP_CH.BAT and DISP.BAT are either listed or referenced in Appendix G, and all are stored on the enclosed diskette in directory APPNDX_G.

These programs interact in the following way: The program DISP is invoked from directory \FLUX\EXE. It copies file DISP_CH.BAT to disk D (the RAM disk), then moves to D as the default drive, and finally invokes DISP_CH, which copies READP.EXE, CC.EXE, and DISP_CH1.EXE to drive D. DISP_CH then invokes READP, which reads the model parameter file P000.DAT to get the panel amplifier gains (PAG). Next it goes into an infinite loop of repeatedly executing CC and DISP_CH1. Program CC simply digitizes the 16 channels and stores the results in file FIDAT. Program DISP_CH1 reads file FIDAT, converts the data into engineering units (using the panel amplifier gains), and writes the entire table including channel names, voltages, and engineering-unit values to the screen. Thus continuously looping through the programs CC and DISP_CH1 keeps the screen continuously updated. (The reason for operating out of RAM disk is to speed up the process so that the screen is updated approximately once every second rather than much more slowly.)

3.2 Operating Procedures

To begin the real-time tabular display:

> Type DISP and then press Enter.

To stop the screen display:

> Hold down the Control Key and type C. (You may have to repeat this several times.) When the message "Terminate batch job (Y/N)?" appears, simply type Y and this will return you to the default drive D. At this point you will normally want to type C: and press Enter, thus returning to directory C:\FLUX\EXE.

4. DATA ACQUISITION

The Data Acquisition module (Fig. 1, box S2), working in conjunction with an ADC board in the PC, digitizes 16 channels of analog data and creates two data files for each run.

4.1 Organization of Program Files

The operational program files for the data acquisition module are FLUXDAT.EXE and GETHFF.EXE, and their respective procedure files are FLUXD.BAT and GET-DAT.BAT. These four files are located in directory \FLUX\EXE on the WIN AT-clone field computer. The source files for the main programs, FLUXDAT.C, VP.H, and GETHFF.FOR, and the procedure files are partially listed in Appendix H and stored in full on the enclosed diskette in directory APPNDX_H.

The program FLUXDAT performs the digitizing and creates the data files Lxxx.DAT and Hxxx.DAT. At this point, however, the high-frequency data file Hxxx.DAT is left on the D drive (RAM disk) and in an incorrect format. The program GETHFF (GET High Frequency File) "rescues" the file from the volatile RAM disk, performs the necessary reformatting and some other corrections (see comments in the source file GETHFF.FOR in Appendix H), and copies the transformed file to a non-volatile disk.

4.2 Operating Procedures

To make a data run we follow the steps outlined below:

(1) Go to directory \FLUX\EXE on the field computer.

(2) Type the command FLUXD xxx (where xxx is the three-digit run number) and then press Enter. This deletes any high-frequency data file from the RAM disk and invokes the FLUXDAT program. FLUXDAT performs an initialization and then arms the system to begin digitizing when the next trigger pulse arrives from the trigger/clock pulse generator. (A trigger pulse occurs every 40.96 seconds.) The digitizing will begin and then continue through 32 trigger-pulse periods (1310.72 seconds). At the end of the run the DOS operating-system prompt will return to the screen.

(3) Insert a formatted 1.2 Mbyte diskette into drive A, type GETDAT xxx and press Enter. (Again xxx is the three-digit run number.) This procedure file first invokes GETHFF, which reformats file Hxxx.DAT and outputs it to drive A, then it copies file Lxxx.DAT to drive A.

(4) Remove the diskette from drive A, and label it with the run number.

5. GRAPHIC DATA DISPLAY

The Graphic Data Display (Fig. 1, box S3) is a support module used to display the data files graphically in a time-series format. This module is particularly useful for verifying the correct operation of a particular instrument, or for explaining an unusual value of a meteorological parameter derived from one or more data channels.

The display format is such that two channels are displayed simultaneously on the screen—one above the other. Any two of the high-frequency channels or any two of the low-frequency channels may be paired together. This gives the opportunity to compare the temporal relationship between the responses of different instruments.

I have not yet made a comprehensive visual examination of the data, but my colleagues and I have confirmed our suspicions about certain portions of the data. I confirmed that the CN2LO and CN2HI channels were not working for runs 3–7. We also confirmed, from a small sampling, that all the runs that we checked in the range 1–11 contained at least some spikes on the sonic anemometer channels, but that none of the runs that we checked in the range 12–52 contained any spikes on the sonic channels. This agrees with our oscilloscope observations during the experiment that the installation of a second power-line filter after run 11 seemed to cure the interference problem of the sonic anemometer. (Additional work should be done to estimate the degree of degradation caused by the spikes and whether or not some remedial action is needed.)

5.1 Organization of Program Files

The operational program files for the graphical data display are DSHI.EXE and DSLO.EXE, and their respective procedure files are DSH.BAT and DSL.BAT. These four files are located in directory \FLUX\EXE on the analysis computer. Program DSHI displays the high-frequency channels (the first 4 channels), whereas program DSLO displays the low-frequency channels (the remaining 12 channels).

The source files for the high-frequency display program DSHI are DSHI.C, YPIX.C, MAXAB.C, and GULP.C plus three other files READB.C, RECT.C, and TMSHFT.C that are used in common with the low-frequency program. The source files for the low-frequency display program DSLO are DSLO.C, YPIXN.C, MINMAX.C, and DSET.C plus the three common files. These source files along with the procedure files DSH.BAT and DSL.BAT, and the "make" files DSHI.MK and DSLO.MK are given in part in Appendix I and in whole in directory APPNDX_I of the enclosed diskette.

5.2 Operating Procedures for Displaying High-Frequency Data

To display two of the high-frequency channels from a data run we need the data file Hxxx.DAT, as indicated in Fig. 1. (The xxx represents the three-digit run number.) We illustrate, using run number 25, the procedures for displaying the data:

(1) Go to directory \FLUX\EXE on the analysis computer by typing the command CD\FLUX\EXE.

(2) Insert the diskette for run 25 into drive A.

(3) Execute the command DSH 025. This procedure file first executes the command MSHERC. (This is a command supplied as part of the Microsoft C 5.1 Optimizing Compiler for configuring graphics programs for the Hercules monographics standard.) Next, it deletes any previous data files from the F: drive (the RAM disk), and then it copies the file H025.DAT from the A: drive to the F: drive. Finally it invokes the main program DSHI.

(4) At this point the program will ask for a "Time-scale compression factor." Respond with a 1 if you want no time-scale compression (one pixel per data point), or respond with another number, ranging from 2 to 50, if you want some time-scale compression. (A compression factor of 50 puts the entire data run on the screen at one time.) The program will next ask you for a channel number. Respond with a 1, 2, 3, or 4 to choose the sonic anemometer channel A, B, or W, or the temperature channel T', respectively, as the upper graph on the screen. The same question will be repeated for the lower graph. After you respond, the graphs will appear.

Following is a brief explanation of what you will see and how to manipulate the graph: You will see two rectangles, each containing a time trace. At the left of each rectangle are three numbers. The upper and lower numbers are the upper and lower bounds of the graph in engineering units—meters per second for the first three channels and degrees Celsius for the fourth channel. Between these two numbers is the channel number, followed by a channel symbol in parentheses. The numbers along the bottom of the lower graph show the time in seconds.

To manipulate the time trace, use the right and left arrow keys to move the trace forward and backward, respectively.

Various menu items at the bottom of the screen give additional flexibility for viewing the data. If the trace goes off scale, typing an N turns on the Auto-normalization mode, which automatically rescales the data when needed. Typing a C allows you to change the Compression factor (time scale).

If you desire to move through the data with a larger or smaller time shift for each press of an arrow key, you can type in a new time (in seconds) and press Enter. To query the program for its current status, type a question mark (?), and the current Time shift, Compression factor, and Auto-normalization mode will be displayed. Finally, typing a Q allows you to Quit (exit) the program.

5.3 Operating Procedures for Displaying Low-Frequency Data

The operation of program DSLO for displaying low-frequency data from run 25 is just a little different. Steps one and two are the same as those in Section 5.2. Steps three and four are indicated below:

(3) Execute the command DSL 025 . The low-frequency data file is sufficiently small that copying it to the RAM drive offers no advantage. Otherwise this procedure file acts very similar to its high-frequency counterpart.

(4) The program continues to parallel the high-frequency version except that the time compression range is now 1 to 4 and the low-frequency channel numbers range from 1 to 12. These channels correspond to the following:

```
 1. CN2LO
 2. CN2HI
 3. CN2LSR
 4. CN2LA
 5. WOPT
 6. CV2
 7. CT2LO
 8. CT2HI
 9. PROP
10. VAN
11. BARM
12. ANGL
```

Acronyms for channels 1–8 are explained in Appendix A; channels 9–12 are for propeller-wind speed, wind direction, barometric pressure, and angle of instrument boom, respectively.

6. DATA ANALYSIS

The purpose of the Data Analysis module (Fig. 1, box S4) is to analyze the data as prescribed in Appendix A of this report, and to output the results into an ASCII file.

6.1 Organization of Program Files

The operational program file FLUX1.EXE (version 1.9, April 20, 1989) as well as the procedure file FLUXA.BAT are in directory \FLUX\EXE on the analysis computer. Because of the size of the program, the source code is contained in eight separate files: FLUX1.FOR, OUT1.FOR, OUT2.FOR,..., OUT7.FOR. The procedure file is listed in Appendix J as well and a portion of FLUX1.FOR. The procedure file and all of the source files are stored in directory APPNDX_J of the enclosed diskette.

6.2 Operating Procedures

To process the data from a field run we need the two data files from that run, Hxxx.DAT and Lxxx.DAT, and the parameter file Pxxx.DAT, as indicated in Fig. 1. (The xxx represents the three-digit run number.) We illustrate the procedures for analyzing a data run using run number 25; these are the steps we need to follow:

(1) Go to directory \FLUX\EXE on the Everex 386/20 computer.

(2) Insert the diskette for run number 25. (This diskette will contain the data files H025.DAT and L025.DAT and the parameter file P025.DAT .)

(3) Type in the command FLUXA 025 and then press Enter. This will invoke the procedure file FLUXA.BAT, which will (1) copy both data files and the parameter file from the diskette to the hard disk, (2) invoke version 1.9 of the analysis program FLUX1.EXE, which will output the analysis file F025.V19, (3) copy F025.V19 to the diskette, and (4) delete the data files (but not the parameter and analysis files) from the hard disk.

(4) Remove the diskette from the machine.

At the end of this process the analyzed data file F025.V19 will have been added to the diskette for run number 25, and that file along with the parameter file will remain on the hard disk in directory \FLUX\EXE.

6.3 Overview of Analysis Program FLUX1

In this brief overview of the analysis program (see Appendix J), I include some of my personal preferences in programming techniques. I do this by going through the main program (about three pages of code after initial comments) in detail. This should give a good sense of the structure of the entire program (approximately 48 pages of code). This procedure, in fact, illustrates the **first** technique for making a program more readable and maintainable: *The overall structure of a program should be given in the main program and individual tasks capsuled into subroutines.* I recommend as a guideline that the main program be from one to four pages of code; if it is more than this it very likely contains too much detail.

A **second** technique, which also helps to communicate the structure, is to *divide the main program into functional blocks, giving a heading to each block.* In the main program FLUX1 (see Appendix J), after about two pages of comments, I divide the program into the following major blocks:

(1) Two blocks of specification statements with the headings

"High freq arrays and parameters"

"Low freq arrays and parameters";

(2) A block that ingests and checks parameters

"Get high and low freq parameters";

(3) Two blocks for computing "primitives" from the time-series data

"Now process high freq data"

"Now process low freq data";

(4) A block for analyzing and outputting all the data

"Now analyze and output data ...."

This final block is divided into sub-blocks, which call the subroutines OUT1, OUT2, ... OUT7 that correspond exactly to the headings in Appendix A of this report.

Ordinarily I would place all of the specification statements into one major block. In this program, however, because of the large number of specification statements, and because the time series were so neatly divided between high-frequency and low-frequency data, it seemed helpful to partition the specification statements into two blocks.

The first executable statement in the main program (immediately before the heading "Get high and low freq parameters") is VER = 1.9. The version number VER is later passed to subroutine OUTHDR where it is written to the header of the output file.

The function of the block of code headed "Get high and low freq parameters" is to get all of the parameters necessary for the data processing. The first subroutine CHKCMD (CHecK CoMmanD) checks for the proper command-line calling sequence, (giving screen prompts if it is incorrect) and outputs the run number in two formats.

The next block of code—headed "Now process high freq data"—processes the high-frequency time-series data. The first statement "WRITE ... " uses the run number in character format IRUNCH to construct the file name HNAME that is used in the OPEN statement. The "WRITE(*,*) ... " statement and subsequent statements of this type simply give progress reports to the screen.

The DO-loop reads and processes one-fourth of the time-series data each time through the loop. The subroutine INHDAT (INput High freq DATa) reads the Ith quarter of each of the data channels and outputs these data points into the integer arrays IA, IB, IW, and IT. These integer arrays are then input to the subroutine ENGH_A or ENGH_B (ENGineering units for High freq data), which converts all the data into engineering units and outputs them as the floating point arrays A, B, W, and T. (Version B of this subroutine is normally used. Version A is used for runs 1 and 2 in which the polarity of the T' sensor was reversed.) The subroutine SONIC1 takes the data A and B from the nonorthogonal arms of the sonic anemometer and converts them into components U (perpendicular to the boom) and V (parallel to the boom). SONIC1 also produces the array HISTA, which contains HISTagram type data of wind direction from the sonic Anemometer. The final subroutine in the loop COMPH (COMPute High frequency data) takes the arrays U, V, W, and T and computes certain "primitives" (consolidated quantities from which standard meteorological parameters can be computed) from the time-series data. For example, UAV(I) is simply the average value of the array U for the Ith quarter, UW(I) is the similar average of the running product of arrays U and W, and SAV(I) is the average of the wind speed (SQRT(U**2+V**2)).

Note at this point that the arrays UAV, VAV, ... are of length five, and that at the completion of the DO-loop only the first four positions of these arrays have been filled. The subroutine COMPH1, outside the loop, computes the overall averages of these quarterly averages and puts them into the fifth position of each of these arrays.

The observant reader might have noticed a "natural" distinction between the input and output parameters in the parameter calling sequences. This distinction is intentional; it makes the program much easier to read. In all the subroutine calls through the end of the loop, all of the **input** parameters are given first, followed by a **space**, and then all of the **output** parameters. For example, the subroutine INHDAT has input parameters IU, I, NHICH, NHPTB, and NBLKS, and output parameters IA, IB, IW, and IT. The input parameters tell the subroutine the unit number of the file to read (IU), which quarter of the data to read (I), the Number of HIgh frequency CHannels (NHICH) in the file, the Number of High frequency PoinTs in one quarterly Block of data (NHPTB), and the Number of such BLocKS (NBLKS) of data. The outputs are the quarterly output arrays IA, IB, ... each containing NHPTB (8192) points. The subroutine ENGH_A or ENGH_B takes these integer data arrays and certain other parameters—including TCBAR, the average Celsius temperature, and PAG, the array of Panel Amplifier Gains—to produce the output arrays A, B, W, and T in engineering units.

The subroutine COMPH1 is a little different in that its parameter list contains two input parameters, a space, then a string of eleven other parameters followed by another space. As we discussed earlier these other parameters are actually five-element arrays (in HISTA the elements themselves are arrays), of which the first four elements have already been computed and the fifth is computed in this subroutine. Thus each of these parameters (array names) actually takes information into the subroutine, via the first four elements, and carries information out of the subroutine, via each of the fifth elements. I call this category input/output parameters and place them in sequence between the pure input and the pure output parameters. Thus, a **third** technique for making a program more readable is to *categorize all of a subprogram's formal parameters into no more than three groups; input, input/output, and output; and then arrange the groups in the above sequence with spaces separating groups.* See Appendix B for more discussion of this technique.

Notice that the final output of this block of code is a set of five-element arrays, UAV, VAV, ..., SAV, and HISTA. The ten arrays UAV through SAV are all combined into the five by ten array DATHI using an EQUIVALENCE statement earlier. This provides a more convenient way to pass these data into some of the output subroutines (OUT1 and OUT6) in the last block of code. Of course, passing the data through a COMMON would be one method of "simplification," but I strongly recommend against this because it effectively hides the data flow from the reader. Thus a **fourth** technique for making programs more readable is to, whenever possible, *avoid the use of COMMONs*. I, in fact, never use COMMONs unless forced to do so because of using a "canned" subroutine that contains a COMMON.

The next block of code—with the heading "Now process low freq data"—is structurally almost identical to the previous block. One small difference is that whereas the prior block output a set of ten five-element primitives that were EQUIVALENCEd to one larger array, the current block simply declares the larger array at the outset. In this case the larger array DATLOW contains thirteen five-element primitives.

The final block of code—with the heading "Now analyze and output data into file Fxxx.Vyy ..." —takes the primitives DATHI and DATLOW, performs the computations prescribed in Appendix A, and outputs the results into an easily human-readable ASCII file. (Appendix F contains a sample file.) As mentioned earlier in this section, the seven output subroutines, OUT1, ..., OUT7, perform the calculations specified in sections A1 through A7 of Appendix A.

One thing slightly unusual in this block is the naming structure of some of the variables, for example, EPS_OPT_RI being output from subroutine OUT2. The data analysis prescribed in Appendix A contains many variables with multiple subscripts. Therefore I used the underscore _ as a separator between the generic variable name and the first subscript and also between subscripts. I must give a word of caution here; I have violated the ANSI FORTRAN 77 standard in two respects: first, the underscore character is not a legitimate character in ANSI FORTRAN; and second, ANSI FORTRAN only allows six characters in a variable name. The six-character limit is too restrictive, in my opinion, and in fact most FORTRAN compilers allow more than this. (On the PC, Ryan-McFarland FORTRAN Ver. 2.43 allows 31 characters, and Microsoft FORTRAN Ver. 4.01 allows 31 characters if the $NOTRUNCATE metacommand is set.) As for the underscore character, it is allowed by Ryan-McFarland, but not by Microsoft; thus, I used Ryan-McFarland FORTRAN for this project. My general recommendation is this: it is permissible to use variable names of more than six characters, but only use the underscore under unusual circumstances. In either case, know the limits of your FORTRAN compiler.

7. Acknowledgments

No project of this magnitude could be successful without the help and close cooperation of many people. This project is certainly no exception. The person who conceived this experiment and was the driving force behind it was Reginald Hill, who wrote the specifications for the data analysis in Appendix A. Also indispensable to this project was Gerard Ochs, who over the years has designed a steady stream of remote-sensing instruments that exploit the optical-refractive-index inhomogeneities of the atmosphere. Some of these instruments, including his recently developed inner-scale meter form the basis of this experiment. His contribution did not stop there, however; he also spent many long days and nights in the field with me as we made data runs and maintained the equipment. Jim Wilson of our instrument support group, along with his associates Duane Hazen, Scott Abbott, and Mark Kaczar, built the optical instruments and then installed all of the instruments at the field site. John Gaynor and Norbert Szczepczynski of the Atmospheric Studies program area of WPL supplied the propvane anemometer. Dave Call of Atmospheric Instrumentation Research Inc. graciously donated his company's services as he along with Lewis Beeken calibrated the atmospheric pressure sensor. Ray Harrison wrote the data acquisition programs CC.C and FLUXDAT.C, and Robert and Ann Lawrence wrote the graphics display programs DSHI.C and DSLO.C .

This task was supported in part by Atmospheric Sciences Laboratory of White Sands Missile Range through MIPR number ASL 86-8004.

8. REFERENCES

Hill, R.J., W.P. Schoenfeld, J.P. Riley, J.T. Priestley, S.F. Clifford, S.P. Eckes, R.A. Bohlander, and R.W. McMillan (1985): Data analysis of the NOAA-GIT Millimeter-Wave Propagation Experiment near Flatville, Illinois. NOAA Tech. Rep. ERL 429-WPL 60, NOAA Wave Propagation Laboratory, Boulder, CO, 55 pp.

Kaimal, J.C. (1980a): Sonic anemometers. In *Air-Sea Interaction*. F. Dobson, L. Hasse, and R. Davis (Eds.), Plenum Publishing Corp., New York, 81-96.

Kaimal, J.C. (1980b): BAO sensors for wind, temperature, and humidity profiling. The Boulder Low-Level Intercomparison Experiment, Preprint of WMO Report, J.C.

Kaimal, H.W. Baynton, and J.E. Gaynor (Eds.), Rep. No. 2, pp. 1–6. Available from NOAA/ERL, Boulder, CO 80303, and from the NCAR Publications Office, P.O. Box 3000, Boulder, CO 80307.

Kohsiek, W. (1982): Measuring C_T^2, C_Q^2, and C_{TQ} in the unstable surface layer, and relations to the vertical fluxes of heat and moisture. *Boundary-Layer Meteorol.*, **24**, 89–107.

McCracken, D.D., and G. M. Weinberg (1972): How to write a readable FORTRAN program. *Datamation*, **18**(10), 73–77.

Pressman, R.S. (1987): *Software Engineering: A Practitioner's Approach*, Second Edition. McGraw-Hill, New York, 567 pp.

Wyngaard, J.C., Y. Izumi, and S.A. Collins, Jr. (1971): Behavior of the refractive-index structure parameter near the ground. *J. Opt. Soc. Am.*, **61**, 1646–1650.

APPENDIX A

SPECIFICATIONS FOR DATA ANALYSIS

This appendix specifies the data analysis performed by the Data Analysis module indicated by box S4 of Fig. 1 and described in Section 6. The analysis program FLUX1 calls seven major subroutines OUT1, OUT2, ... OUT7. These subroutines perform the analysis described by Sections A1 through A7, respectively, of this appendix.

Understanding this appendix requires the following notation. Where two symbols are used, the first is the conventional scientific symbol and the second is the FORTRAN variable name used in the processing software. Angle brackets denote a time average over the duration of the data run.

$C_n^2(Z_L)$ CN2LO	Refractive-index structure parameter from a large-aperture scintillometer on a 607–m path at height Z_L, where Z_L = 1.45 m. Radiation wavelength λ_L = 0.94 μm; diameter $D_L \simeq$ 0.146 m.
$C_n^2(Z_H)$ CN2HI	Refractive-index structure parameter from a large-aperture scintillom- on the same (607–m) path at height Z_H, where Z_H = 3.95 m. Radiation wavelength $\lambda_H = \lambda_L$; diameter $D_H = D_L$.
$C^2_{n,\text{Laser}}$ CN2LSR	Scaled log-intensity variance from laser scintillation on a path of length L = 150 m having wavelength $\lambda = 0.628 \times 10^{-6}$ m and using a diverged wave, at the height Z_H.
$C^2_{n,\text{LA}}$ CN2LA	Scaled log-intensity variance from a large-aperture scintillometer of diameter D_{LA} = 0.044 m, on the same path, of length L and height Z_H, as the laser. Wavelength of radiation is λ_{LA} = 0.94 μm.
$C_T^2(Z_L)$ CT2LO	Temperature structure parameter measured by two platinum resistance wires spaced vertically by the distance R_T = 0.1 m and centered at height Z_L (the same height as the lower large-aperture scintillometer).
$C_T^2(Z_H)$ CT2HI	Same as $C_T^2(Z_L)$ but at height Z_H (the height of the higher large-aperture scintillometer).
W_{opt} WOPT	Mean vertical wind speed obtained by a large-aperture optical cross-wind scintillometer having its receiver-separation axis mounted vertically. Instrument is set at height Z_H.
C_V^2 CV2	Velocity structure parameter from two hot-film-on-quartz-rod anemometers spaced vertically with separation R_V = 0.1 m and centered at a height Z_H.

P BARM or PBAR	Total atmospheric pressure in millibars.
T	Absolute temperature from a thermometer in kelvin.
$< w'T' >$ TMPFLX	The temperature flux obtained by correlation of vertical velocity fluctuations from the sonic anemometer (referred to simply as the sonic in this list) (w') with the temperature fluctuations (T') from the platinum wire within the sonic.
$< T'^2 >$	Temperature variance from fluctuations of the sonic's fine-wire sensor.
$< u'w' >$ UPWP	Correlation of the sonic's horizontal velocity component straight toward the sonic (transverse to boom), u', with the vertical component from the sonic, w'.
$< v'w' >$ VPWP	Correlation of the sonic's horizontal velocity component orthogonal to u' (parallel to the boom) with the vertical component of the sonic, w'.
$< w'^2 >$	Vertical-velocity variance from the sonic.
$<u>$ U	Mean, toward-boom, horizontal velocity component from the sonic.
$<v>$ V	Mean, parallel-to-boom, horizontal velocity component from the sonic.
$<w>$ W	Mean vertical velocity component from the sonic.
$<s>$ SAV	Mean wind speed from the sonic.

A1. PROCESSING OF SONIC ANEMOMETER AND T′ DATA

The A and B wind-velocity components from the sonic anemometer are described by Kaimal (1980a) and by Hill et al. (1985, pp. 18–22). The u and v components are then as defined in these papers: u is positive into the sonic anemometer and v is positive for wind blowing right to left of a person facing toward the front of the sonic anemometer. The instantaneous angle of the horizontal wind relative to straight into the boom is

$$a = \tan^{-1}(v/u) \;.$$

If a is outside the range $\pm 45°$, the wind measurement is less accurate; thus such occurrences should be monitored. An approximate correction exists for the drag of the sonic arms for angles up to $\pm 60°$ (Kaimal, 1980b). It is obvious that the measurements by the sonic anemometer must be accurately nulled for zero wind if these angles are to be known adequately.

The mean wind speed from the sonic anemometer is obtained from $< s >=< \sqrt{u^2 + v^2} >$. The angle of the vector mean wind from the sonic anemometer can be defined in two ways. The usual definition,

$$\theta_1 = \tan^{-1}(< v > / < u >)$$

assumes the average cross-stream velocity is zero ($< v_{\theta_1} > \; = \; 0$): note that the FORTRAN function ATAN2 is used for $\tan^{-1}$. The second definition,

$$\theta_2 = \tan^{-1}(< v'w' > / < u'w' >)$$

assumes a horizontally homogeneous surface layer where we can set the cross-stream momentum flux to zero ($< v'_{\theta_2} w' > \; = \; 0$) (see Hill et al., 1985). This second definition could be superior in cases where the zero offset of the sonic anemometer may have drifted substantially. Analogous to the first definition, we define a third mean wind direction from the propeller-vane anemometer (propvane) data as

$$\theta_3 = \tan^{-1}(< v_{\mathrm{propvane}} > / < u_{\mathrm{propvane}} >) \;,$$

where $u_{\mathrm{propvane}} = s \, \cos\phi$ and $v_{\mathrm{propvane}} = s \, \sin\phi$, and s and ϕ are the instantaneous wind speed and angle from the propellor-vane anemometer. The vertical flux of horizontal momentum per unit mass density is given by Eq. (61) of Hill et al. (1985):

$$< u_\theta' w' > \; = \; < u'w' > \cos\theta + < v'w' > \sin\theta \;,$$

where θ can be any of θ_1, θ_2, or θ_3. Thus, it can be computed for each of the three angles above. Of course, in forming any correlation like $< u'w' >$ we use $< u'w' > = <uw> - <u><w>$, since, for instance, $u' \equiv u - <u>$.

The cross-stream momentum flux [Eq. (64) in Hill et al., 1985] can similarly be calculated for the three measures of wind angle.

The friction velocity is then defined as

$$u_* = \sqrt{-< u_\theta' w' >} \ .$$

The temperature scaling parameter is defined by

$$T_* = -< T'w' > / u_* \ .$$

The Monin-Obukhov length is defined by

$$L_{\mathrm{MO}} = \frac{u_*^2 T}{gKT_*}$$

where $g = 9.8$ m s^{-2} is the acceleration due to gravity,

and $K = 0.4$ is the VonKarman constant.

A2. INNER SCALE FROM SCINTILLATION AND C_v^2

The inner-scale meter has two outputs, a scaled laser variance and a scaled large-aperture-scintillometer variance. If the inner scale is zero then these scaled variances would be C_n^2, which is why they are sometimes referred to as "C_n^2". These scaled variances, which we shall designate $C_{n,\mathrm{Laser}}^2$ and $C_{n,\mathrm{LA}}^2$ because of their close association with C_n^2, are related to the log-intensity variances by

$$C_{n,\mathrm{Laser}}^2 = \sigma_{\mathrm{Laser}}^2/(0.5k^{7/6}L^{11/6})$$
(CN2LSR)

where $k = 2\pi/\lambda$ and L is propagation path length, and

$$C_{n,\mathrm{LA}}^2 = \sigma_{\mathrm{LA}}^2/(0.9D_{\mathrm{LA}}^{-7/3}L^3)$$

(CN2LA)

where D_{LA} is the diameter of the large-aperture scintillometer. The FORTRAN variables are shown in parentheses and $\sigma^2_{\mathrm{Laser}}$ and σ^2_{LA} are the log-intensity variances of the instruments. This scaling is provided by potentiometer settings.

After averaging for 20 minutes, we calculate the ratio

$$y = C^2_{n,\mathrm{Laser}}/C^2_{n,\mathrm{LA}} \cdot$$

Using y, Table A1 allows determination of the inner scale, ℓ_{opt} (L_OPT); Table A1 is incorporated in subroutine OUT2, and the interpolation is performed in subroutine OUT2B. The correction for dispersion is already incorporated in Table A1. This table is specific to the wavelengths and propagation pathlengths of our experiment; a different table is needed for different propagation pathlengths or for a different laser wavelength.

Table A1. Values for determing inner scale from y, and for determining C_n^2 from the variances obtained from large-aperture and laser scintillometers.

y	Inner scale (m)	SLAP	SLASR
1.1398108	0.0005000	1.0000000	1.1220667
1.1474419	0.0005211	1.0000000	1.1295790
1.1553192	0.0005431	1.0000000	1.1373337
1.1634406	0.0005660	1.0000000	1.1453286
1.1717749	0.0005899	1.0000000	1.1535333
1.1802753	0.0006149	1.0000000	1.1619013
1.1888820	0.0006408	1.0000000	1.1703740
1.1975204	0.0006679	1.0000000	1.1788779
1.2061170	0.0006961	1.0000000	1.1873407
1.2146411	0.0007255	1.0000000	1.1957321
1.2230703	0.0007561	1.0000000	1.2040300
1.2313820	0.0007880	1.0000000	1.2122124
1.2395538	0.0008213	1.0000000	1.2202569
1.2475506	0.0008560	1.0000000	1.2281292
1.2552377	0.0008921	1.0000491	1.2357575
1.2624282	0.0009298	1.0002233	1.2430528
1.2691200	0.0009691	1.0004395	1.2499120
1.2751852	0.0010100	1.0007026	1.2562158
1.2805026	0.0010526	1.0010182	1.2618518
1.2849981	0.0010971	1.0013921	1.2667549
1.2886058	0.0011434	1.0018310	1.2708681
1.2912623	0.0011917	1.0023421	1.2741377
1.2929084	0.0012420	1.0029334	1.2765145
1.2934813	0.0012944	1.0036136	1.2779463
1.2929050	0.0013491	1.0043925	1.2783683
1.2911047	0.0014060	1.0052806	1.2777171
1.2880102	0.0014654	1.0062897	1.2759341
1.2835580	0.0015273	1.0074326	1.2729678

Table A1. (continued)

y	Inner scale (m)	SLAP	SLASR
1.2776826	0.0015918	1.0087235	1.2687646
1.2703073	0.0016590	1.0101779	1.2632596
1.2613609	0.0017290	1.0118130	1.2563931
1.2507818	0.0018020	1.0136476	1.2481146
1.2385215	0.0018781	1.0157024	1.2383856
1.2245324	0.0019574	1.0180001	1.2271680
1.2087604	0.0020401	1.0205659	1.2144152
1.1911648	0.0021262	1.0234274	1.2000928
1.1717245	0.0022160	1.0266149	1.1841834
1.1504415	0.0023096	1.0301617	1.1666910
1.1273295	0.0024071	1.0341047	1.1476284
1.1024089	0.0025087	1.0384843	1.1270121
1.0757298	0.0026147	1.0433450	1.1048850
1.0473791	0.0027251	1.0487359	1.0813243
1.0174876	0.0028401	1.0547074	1.0564453
0.9861764	0.0029600	1.0612838	1.0303199
0.9535165	0.0030850	1.0684712	1.0029447
0.9195956	0.0032153	1.0762674	0.9743231
0.8845242	0.0033511	1.0846601	0.9444725
0.8484395	0.0034926	1.0936251	0.9134299
0.8115563	0.0036400	1.1031229	0.8813095
0.7741817	0.0037937	1.1130962	0.8483236
0.7366788	0.0039539	1.1234662	0.8147495
0.6994704	0.0041209	1.1341281	0.7809394
0.6629960	0.0042949	1.1449465	0.7472777
0.6274534	0.0044762	1.1557497	0.7138897
0.5929258	0.0046652	1.1663282	0.6807803
0.5593825	0.0048622	1.1766185	0.6479335
0.5266531	0.0050675	1.1868107	0.6153072
0.4945433	0.0052815	1.1971731	0.5828372
0.4629605	0.0055045	1.2080508	0.5505732
0.4319554	0.0057369	1.2197592	0.5186793
0.4017687	0.0059791	1.2321817	0.4873453
0.3726914	0.0062316	1.2450264	0.4567871
0.3450124	0.0064947	1.2579160	0.4272403
0.3189241	0.0067689	1.2703722	0.3988450
0.2945263	0.0070548	1.2818326	0.3716561
0.2717732	0.0073526	1.2921441	0.3457033
0.2505134	0.0076631	1.3015731	0.3209855
0.2305825	0.0079867	1.3104887	0.2974716
0.2119094	0.0083239	1.3190809	0.2751741
0.1945038	0.0086754	1.3271861	0.2541240
0.1783564	0.0090417	1.3345792	0.2343252
0.1634295	0.0094234	1.3409903	0.2157456
0.1496600	0.0098213	1.3460980	0.1983208
0.1370233	0.0102360	1.3495208	0.1820371
0.1254964	0.0106682	1.3509275	0.1668972
0.1149511	0.0111187	1.3509304	0.1528734

Table A1. (continued)

y	Inner scale (m)	SLAP	SLASR
0.1052436	0.0115882	1.3503098	0.1398991
0.0962922	0.0120775	1.3490059	0.1278765
0.0880726	0.0125874	1.3468289	0.1167721
0.0805746	0.0131189	1.3435532	0.1065710
0.0737669	0.0136729	1.3389120	0.0972298
0.0675917	0.0142502	1.3325903	0.0886698
0.0619731	0.0148519	1.3242195	0.0807884
0.0568825	0.0154790	1.3135066	0.0735524
0.0522972	0.0161326	1.3003700	0.0669471
0.0481799	0.0168138	1.2847767	0.0609368
0.0444745	0.0175237	1.2667294	0.0554601
0.0411129	0.0182637	1.2462765	0.0504404
0.0380687	0.0190348	1.2235214	0.0458528
0.0353247	0.0198386	1.1986359	0.0416823
0.0328516	0.0206762	1.1718738	0.0378986
0.0306033	0.0215493	1.1435881	0.0344528
0.0285248	0.0224592	1.1142378	0.0312887
0.0266079	0.0234075	1.0841059	0.0283967
0.0248618	0.0243958	1.0532230	0.0257774
0.0232912	0.0254259	1.0216142	0.0234242
0.0218931	0.0264995	0.9893121	0.0213219
0.0206535	0.0276184	0.9563578	0.0194446
0.0195427	0.0287846	0.9228023	0.0177533
0.0185091	0.0300000	0.8887078	0.0161931

If the inner scale is zero and the saturation of scintillation is negligible, and the diverged laser beam can be approximated as initially a spherical wave, and the true refractive-index structure parameter in the atmosphere C_n^2, is uniform along the propagation path, then the Rytov variance of log-amplitude β_o^2, is given by

$$\beta_o^2 \equiv 0.124 k^{7/6} L^{11/6} C_n^2 \ .$$

Now assume all of the above restrictions with the exception that the inner scale may be nonzero. To obtain the Rytov log-amplitude variance including the inner-scale effect σ_R^2, we read from Table A1 the value of SLASR that corresponds to a given value of inner scale, then

$$\sigma_R^2 = \beta_o^2 * \text{SLASR} \ .$$

That is, for a given inner scale SLASR is the proportionality constant between σ_R^2 and β_o^2 for our experimental conditions.

We need to check the effect of saturation of scintillation on the laser variance. To do this we first calculate the log-amplitude variance

$$\sigma^2_{\chi,\mathrm{Laser}} = 0.124 k^{7/6} L^{11/6} C^2_{n,\mathrm{Laser}}$$

(SIG2CHI)

and $\sqrt{\lambda L}/\ell_{\mathrm{opt}}$. With these values we use an interpolation over Table A2 to determine a value of the Rytov variance that includes inner scale effect, σ_R^2. Table A2 and the interpolation routine are incorporated in subroutine RYTOV4. RYTOV4 is called from subroutine OUT2. The data statement representing Table A2 in RYTOV4 contains some extrapolated values as well as the values in Table A2. To use Table A2 we locate the appropriate value of $\sqrt{\lambda L}/\ell_{\mathrm{opt}}$ at the top and read down the column to a value that matches the value of $\sigma^2_{\chi,\mathrm{laser}}$ then over to the left to get a value of β_o^2. Note that for $\sqrt{\lambda L}/\ell_{\mathrm{opt}} < 1.0$ one should use the column for $\sqrt{\lambda L}/\ell_{\mathrm{opt}} = 1.0$. We obtain SLASR corresponding to ℓ_{opt} from Table A1. Then we obtain $\sigma_R^2 = \beta_o^2 * \mathrm{SLASR}$. If σ_R^2 differs substantially from $\sigma^2_{\chi,\mathrm{Laser}}$, then we correct our ratio as follows:

$$C^2_{n,\mathrm{Laser,RI}} = \sigma_R^2/(0.124 k^{7/6} L^{11/6})$$

$$y_{\mathrm{RI}} = C^2_{n,\mathrm{Laser,RI}}/C^2_{n,\mathrm{LA}} \, .$$

These are approximately the values that would be measured if the Rytov approximation (including the inner-scale effect) was valid for a given data run, that is, if the effects of saturation of scintillation had been negligible. The subscripts RI are a mnemonic for the words Rytov and Inner scale. Note that $C^2_{n,\mathrm{Laser,RI}}$ should always be greater than $C^2_{n,\mathrm{Laser}}$. This corrected y can be used in Table A1 to obtain a corrected value of the inner scale, $\ell_{\mathrm{opt,RI}}$.

We now determine C_n^2 from the optical inner-scale meter. There are two values, one from the laser and one from the large-aperture scintillometer. The values of SLAP and SLASR are obtained from Table A1 for the determined inner scale ℓ_{opt}. This means using $\ell_{\mathrm{opt,RI}}$ if the saturation of scintillation was significant. From the large-aperture scintillometer the value of C_n^2 is

$$C^2_{n,\mathrm{LA,RI,IS}} = C^2_{n,\mathrm{LA}}/\mathrm{SLAP}_{\mathrm{RI}}$$

Table A2. Values of $\sigma^2_{\chi,\mathrm{Laser}}$ for given values of $\sqrt{\lambda L}/\ell_{\mathrm{opt}}$ and β_o^2.

β_o^2 \ $\dfrac{\sqrt{\lambda L}}{\ell_{\mathrm{opt}}}$	1.0	2.0	4.0	7.0	20.0	∞
0.1	0.0203	0.0639	0.108	0.119	0.1035	0.0919
0.3	0.0602	0.183	0.288	0.302	0.264	0.241
0.6	0.119	0.342	0.486	0.480	0.422	0.397
1.0	0.194	0.523	0.654	0.607	0.536	0.523
1.5	0.284	0.704	0.770	0.668	0.591	0.601
2.0	0.369	0.848	0.818	0.687	0.608	0.629
3.5	0.600	1.105	0.832	0.673	0.596	0.627
5.0	0.798	1.210	0.795	0.655	0.576	0.597
6.5	0.966	1.245	0.769	0.645	0.562	0.575
8.0	1.109	1.231	0.753	0.635	0.553	0.559
10.0	1.269	1.191	0.737	0.627	0.543	0.542
12.0	1.397	1.147	0.725	0.619	0.535	0.534

where the saturation-corrected inner scale, $\ell_{\mathrm{opt,RI}}$, is used to find $\mathrm{SLAP_{RI}}$. From the laser the value of C_n^2 is

$$C^2_{n,\mathrm{Laser,IS}} = C^2_{n,\mathrm{Laser}}/\mathrm{SLASR}$$

where the uncorrected ℓ_{opt} is used to find SLASR. We also calculate a saturation-corrected C_n^2 from the laser using

$$C^2_{n,\mathrm{Laser,RI,IS}} = C^2_{n,\mathrm{Laser,RI}}/\mathrm{SLASR_{RI}}$$

where $\ell_{\mathrm{opt,RI}}$ is used to find $\mathrm{SLASR_{RI}}$. Note that SLASR and SLAP change their values given a change of inner scale due to the saturation of scintillation effect.

Given $\ell_{\mathrm{opt,RI}}$ we correct $C_n^2(Z_H)$ for the inner-scale effect. The values of SLAPL are obtained from Table A3 by interpolation on the inner-scale column. Then the corrected $C_n^2(Z_H)$, $C^2_{n,\mathrm{IS,RI}}(Z_H)$, is obtained using

$$C^2_{n,\mathrm{IS,RI}}(Z_H) = C_n^2(Z_H)/\mathrm{SLAPL}\ .$$

Table A3 is in subroutine OUT2, and the interpolation routine is called in subroutine OUT2B.

An alternative inner scale ℓ_v from the hot-wire anemometers is defined later in this section. By the same method described above, we use Table A3 and ℓ_v (instead of ℓ_{opt} or $\ell_{\text{opt,RI}}$) to obtain a different correction to $C_n^2(Z_H)$, which we designate $C^2_{n,\text{IS},v}(Z_H)$.

The values of $C^2_{n,\text{LA,RI,IS}}$, $C^2_{n,\text{Laser,RI,IS}}$, and $C^2_{n,\text{IS,RI}}(Z_H)$ should all be equal; we need to check this by printing them, and their ratios:

$$\frac{C^2_{n,\text{Laser,RI,IS}}}{C^2_{n,\text{LA,RI,IS}}} \quad \text{and} \quad \frac{C^2_{n,\text{IS,RI}}(Z_H)}{C^2_{n,\text{LA,RI,IS}}} \; .$$

Note that our methods force the former ratio to be unity (actually 1.016 because of dispersion).

Table A3. Values to use for the inner-scale correction to the 0.146 m-diameter large-aperture scintillometer

Inner scale (m)	SLAPL
0.0005000	1.0000000
0.0005211	1.0000000
0.0005431	1.0000000
0.0005660	1.0000000
0.0005899	1.0000000
0.0006149	1.0000000
0.0006408	1.0000000
0.0006679	1.0000000
0.0006961	1.0000000
0.0007255	1.0000000
0.0007561	1.0000000
0.0007880	1.0000000
0.0008213	1.0000000
0.0008560	1.0000000
0.0008921	1.0000000
0.0009298	1.0000000
0.0009691	1.0000000
0.0010100	1.0000000
0.0010526	1.0000000
0.0010971	1.0000000
0.0011434	1.0000000
0.0011917	1.0000000
0.0012420	1.0000000
0.0012944	1.0000000
0.0013491	1.0000000
0.0014060	1.0000000
0.0014654	1.0000000
0.0015273	1.0000000
0.0015918	1.0000000

Table A3. (continued)

Inner scale (m)	SLAPL
0.0016590	1.0000000
0.0017290	1.0000000
0.0018020	1.0000000
0.0018781	1.0000000
0.0019574	1.0000000
0.0020401	1.0000000
0.0021262	1.0000000
0.0022160	1.0000000
0.0023096	1.0000000
0.0024071	1.0000000
0.0025087	1.0000000
0.0026147	1.0000000
0.0027251	1.0000000
0.0028401	1.0000000
0.0029600	1.0000489
0.0030850	1.0002230
0.0032153	1.0004391
0.0033511	1.0007022
0.0034926	1.0010177
0.0036400	1.0013915
0.0037937	1.0018303
0.0039539	1.0023413
0.0041209	1.0029324
0.0042949	1.0036125
0.0044762	1.0043912
0.0046652	1.0052792
0.0048622	1.0062881
0.0050675	1.0074308
0.0052815	1.0087214
0.0055045	1.0101756
0.0057369	1.0118104
0.0059791	1.0136446
0.0062316	1.0156990
0.0064947	1.0179964
0.0067689	1.0205618
0.0070548	1.0234228
0.0073526	1.0266097
0.0076631	1.0301560
0.0079867	1.0340983
0.0083239	1.0384772
0.0086754	1.0433371
0.0090417	1.0487272
0.0094234	1.0546978
0.0098213	1.0612733
0.0102360	1.0684597
0.0106682	1.0762549

Table A3. (continued)

Inner scale (m)	SLAPL
0.0111187	1.0846468
0.0115882	1.0936109
0.0120775	1.1031079
0.0125874	1.1130806
0.0131189	1.1234500
0.0136729	1.1341116
0.0142502	1.1449298
0.0148519	1.1557332
0.0154790	1.1663122
0.0161326	1.1766028
0.0168138	1.1867950
0.0175237	1.1971569
0.0182637	1.2080335
0.0190348	1.2197406
0.0198386	1.2321622
0.0206762	1.2450066
0.0215493	1.2578964
0.0224592	1.2703537
0.0234075	1.2818159
0.0243958	1.2921290
0.0254259	1.3015591
0.0264995	1.3104753
0.0276184	1.3190680
0.0287846	1.3271741
0.0300000	1.3345685

In the same manner, we need to correct $C_n^2(Z_L)$ for the inner-scale effect. To do so we first need the values of the inner scale from optical scintillation and the hot wire-anemometer as extrapolated to the lower height, they are denoted ℓ_{Lo} and ℓ_{Lv}. (The output values are labeled l_L_opt_RI_x, and l_L_v_x where the x is either a 1 or 2 indicating use of the Monin-Obukhov length derived using θ_1 or θ_2.)

To make an inner-scale correction to $C_n^2(Z_L)$ we want to calculate

$$\ell_{Lo} = \ell_{\text{opt}} \left[\frac{Z_L h(\zeta_H)}{Z_H h(\zeta_L)} \right]^{1/4}$$

and

$$\ell_{Lv} = \ell_v \left[\frac{Z_L h(\zeta_H)}{Z_H h(\zeta_L)} \right]^{1/4} .$$

Here, ζ_L and ζ_H are Z_H/L_{MO} and Z_L/L_{MO}, respectively, and L_{MO} is the Monin-Obukhov length from the sonic anemometer data. Also,

$$h(x) = \begin{cases} [1 + 0.46(-x)^{2/3}]^{3/2} & \text{for } x \leq 0 \\ (1 + 2.3x^{3/5})^{3/2} & \text{for } 0 \leq x\ . \end{cases}$$

We also want to compare the energy dissipation rates ϵ_v and ϵ_{opt}, where $\epsilon_{\mathrm{opt}} \equiv \nu^3(7.4/\ell_{\mathrm{opt}})^4$ and ν is kinematic viscosity. We want to perform a scatter plot of both

$$\frac{\epsilon_v K Z_H}{u_*^3} \text{ and } \frac{\epsilon_{opt} K Z_H}{u_*^3} \text{ (NDEDR\_v\_x and NDEDR\_opt\_x in the output)}$$

versus Z_H/L_{MO} and compare it with the function $h(Z_H/L_{\mathrm{MO}})$.

C_v^2 and ϵ_v are related by

$$C_v^2 = (8/3)\epsilon_v^{2/3}\ ,$$

so

$$\epsilon_v = (3C_v^2/8)^{3/2}\ .$$

The coefficient of viscosity is given by

$$\mu = \beta T^{3/2}/(T + S)$$

where $\beta \equiv 1.458 \times 10^{-6}\mathrm{kg\ s^{-1}K^{-1/2}m^{-1}}$ and $S \equiv 110.4\ \mathrm{K}$.

The mass density of air is given by

$$\varrho = M_o P/RT$$

where P is in N m^{-2} , $R = 8.31432 \times 10^3\mathrm{J\ K^{-1}\ kmol^{-1}}$, and $M_o = 28.9644\ \mathrm{kg\ kmol^{-1}}$.

That is,

$$\varrho = P/(287.05T) \ .$$

The kinematic viscosity is given by

$$\nu = \mu/\varrho \ .$$

The inner scale is then obtained from

$$\ell_\nu = 7.4(\nu^3/\epsilon_\nu)^{1/4} \ .$$

The value of ℓ_ν is to be compared with ℓ_{opt}.

A3. COMPARISON OF LARGE-APERTURE C_n^2 VALUES WITH SPACED-WIRE C_T^2

The value of C_T^2 derived from a scintillometer is

$$C_{T,\text{opt}}^2 = C_n^2 10^{12} T^4/[2.842 \times 10^{-3} P N_o(\lambda_H)]^2 = C_n^2 10^{12} (T^2/0.78P)^2 \ ,$$

where P is in N m^{-2} and $N_o(\lambda_H)$ is the refractivity at the wavelength of the scintillometer.

We want to print the ratio of wire to scintillometer-derived C_T^2 for heights Z_H and Z_L. Thus we determine whether or not $C_T^2/C_{T,\text{opt}}^2$ is nearly unity at both Z_H and Z_L.

Also, we want to perform a scatter plot of

$$\frac{Z_L^{2/3} C_T^2(Z_L)}{T_*^2} \text{ versus } Z_L/L_{\text{MO}}$$

$$\frac{Z_L^{2/3} C_{T,\text{opt}}^2(Z_L)}{T_*^2} \text{ versus } Z_L/L_{\text{MO}}$$

$$\frac{Z_H^{2/3} C_T^2(Z_H)}{T_*^2} \text{ versus } Z_H/L_{\text{MO}}$$

$$\frac{Z_H^{2/3} C_{T,\text{opt}}^2(Z_H)}{T_*^2} \text{ versus } Z_H/L_{\text{MO}}$$

and compare these with $G(Z_L/L_{MO})$ and $G(Z_H/L_{MO})$, where several forms for the function $G(x)$ are given in Section A7. In the output the above four quantities are designated NDIM_CT2_LO, NDIM_CT2_OPT_LO, NDIM_CT2_HI, and NDIM_CT2_OPT_HI where NDIM stands for Non-DIMensional.

A4. STABILITY FROM TWO-HEIGHT C_n^2 AND C_T^2

We first define

$$R_n \equiv (Z_H/Z_L)^{2/3}\, C_n^2(Z_H)/C_n^2(Z_L)$$

$$R_T \equiv (Z_H/Z_L)^{2/3} C_T^2(Z_H)/C_T^2(Z_L)\ .$$

We then execute the following logic for both $R_x = R_n$ and $R_x = R_T$ and compare the results.

If $R_x > 1$ we have stable conditions, and

$$\zeta_x = \left\{\frac{R_x - 1}{2.2(Z_H/Z_L)^{1/3}[1 - (Z_H/Z_L)^{-2/3}R_x]}\right\}^{3/2}$$

and $L_{MOx} = \sqrt{Z_L Z_H}\,/\zeta_x$.

If $R_x < 1$ we have unstable conditions, and

$$\zeta_x = \frac{\sqrt{Z_H/Z_L}\,(R_x^{3/2} - 1)}{6.1[(Z_H/Z_L)R_x^{3/2} - 1]}$$

and $L_{MOx} = \sqrt{Z_L Z_H}\,/\zeta_x$ as before. Now compare values of L_{MOn}, L_{MOT}, and L_{MO}.

We assume the functional form

$$G(\zeta) = \begin{cases} A(1 - B\zeta)^{-2/3} & \text{for } \zeta < 0 \\ C(1 + D\zeta^{2/3}) & \text{for } \zeta > 0\ , \end{cases}$$

Then R_x is a sensitive measure of the coefficients B and D. Let $\zeta = \sqrt{Z_L Z_H}\,/L_{MO}$, where L_{MO} is determined from the sonic anemometer. For all data having $\zeta < 0$ we make a scatter plot of ζ versus the values of

$$B = \frac{\sqrt{Z_H/Z_L}\,(1 - R_x^{3/2})}{\zeta\left[1 - \left(\frac{Z_H}{Z_L}\right)R_x^{3/2}\right]} \; .$$

For $\zeta < 0$, let F_x stand for either F_n or F_T defined by

$$F_n = \left[\frac{Z_H^{2/3}\,C_{T,\mathrm{opt}}^2(Z_H)}{T_*^2}\right]^{-3/2} - \left[\frac{Z_L^{2/3}\,C_{T,\mathrm{opt}}^2(Z_L)}{T_*^2}\right]^{-3/2}$$

$$F_T = \left[\frac{Z_H^{2/3}\,C_T^2(Z_H)}{T_*^2}\right]^{-3/2} - \left[\frac{Z_L^{2/3}\,C_T^2(Z_L)}{T_*^2}\right]^{-3/2} ;$$

then $$A = \left[\frac{B(Z_L - Z_H)\zeta}{F_x\sqrt{Z_H Z_L}}\right]^{2/3} .$$

For all data having $\zeta > 0$ we perform a scatter plot of ζ versus the values of

$$D = \frac{R_x - 1}{\zeta^{2/3}\left(\frac{Z_H}{Z_L}\right)^{1/3}\left[1 - \left(\frac{Z_H}{Z_L}\right)^{-2/3} R_x\right]} \; .$$

For $\zeta > 0$, let E_x stand for either E_n or E_T defined by

$$E_n = \frac{Z_H^{2/3}\,C_{T,\mathrm{opt}}^2(Z_H)}{T_*^2} - \frac{Z_L^{2/3}\,C_{T,\mathrm{opt}}^2(Z_L)}{T_*^2}$$

$$E_n = \frac{Z_H^{2/3}\,C_T^2(Z_H)}{T_*^2} - \frac{Z_L^{2/3}\,C_T^2(Z_L)}{T_*^2} ;$$

then $$C = \frac{E_x}{D\zeta\left[\left(\frac{Z_H}{Z_L}\right)^{1/3} - \left(\frac{Z_L}{Z_H}\right)^{1/3}\right]} .$$

A5. THE MEANING OF OPTICALLY-DERIVED W

We seek to determine if W_{opt} is "Monin-Obukhov similar." That is, we want to determine whether or not W_{opt}/u_* is a function of only ζ_H, where $\zeta_H = Z_H/L_{\text{MO}}$, and u_* is the friction velocity, and u_* and L_{MO} are obtained from the sonic anemometer and its wire thermometer.

We make a scatter plot of W_{opt}/u_* versus ζ_H. If the data are obviously organized, then we fit the data to determine an approximate analytic form for the function. We then investigate if stability ζ_H is more usefully obtained by measuring W_{opt} than by measuring C_n^2 at two heights.

A6. THE MONIN-OBUKHOV SIMILARITY OF $< T'^2 >$ AND $< w'^2 >$

The Monin-Obukhov similarity of $< T'^2 >$ and $< w'^2 >$ is an old topic. We add to it using our own data. We therefore make scatter plots of

$$\frac{< T'^2 >}{T_*^2} \quad \text{and} \quad \frac{< w'^2 >}{u_*^2}$$

versus $\zeta_H = Z_H/L_{\text{MO}}$. T_*, u_*, and L_{MO} are sonic-anemometer and wire-derived quantities.

A7. FLUXES FROM PROPAGATION

Finally, we determine heat and momentum fluxes from propagation statistics. There are three possible formulas for G.

(1) Wyngaard et al. (1971):

$$G(\zeta) = \begin{cases} 4.9(1 + 7.0|\zeta|)^{-2/3} & \zeta \le 0 \\ 4.9(1 + 2.75\zeta^{2/3}) & \zeta \ge 0 \ . \end{cases}$$

(2) Modified Wyngaard et al. [our own fit to the data by Wyngaard et al. (1971)]:

$$G(\zeta) = \begin{cases} 8.1(1 + 15|\zeta|)^{-2/3} & \zeta \le 0 \\ 2.7(1 + 5.0\zeta^{2/3}) & \zeta \ge 0 \ . \end{cases}$$

(3) Kohsiek [our own fit to the data by Kohsiek (1982)]:

$$G(\zeta) = \begin{cases} 17(1+46|\zeta|)^{-2/3} & \zeta \le 0 \\ 2.7(1+5.0\zeta^{2/3}) & \zeta \ge 0\ . \end{cases}$$

Kohsiek's formula for $G(\zeta)$ for $\zeta \geq 0$ is not known, so we take it to be the same as the Modified-Wyngaard et al. formula.

The function h is given by

$$h(\zeta) = \begin{cases} (1+0.46|\zeta|^{2/3})^{3/2} & \zeta \le 0 \\ (1+2.3\zeta^{3/5})^{3/2} & \zeta \ge 0\ . \end{cases}$$

We define the value

$$V \equiv g^2 K^{2/3} Z_H^{4/3}\, C_T^2 / \{T^2 [\epsilon_{\text{opt}}(Z_H)]^{4/3}\},$$

where C_T^2 is obtained from $C_{n,LA,RI,IS}^2$ using the first formula in Sec. A3. We then solve the following formula by iteration, for all three possible forms for $G(\zeta)$:

$$\zeta^2 G(\zeta)[h(\zeta)]^{-4/3} = V\ .$$

The scintillation value of stability is then ζ and the scintillation value of Monin-Obukhov length is then

$$L_S = Z_H/\zeta\ .$$

The scintillation values of friction velocity are

$$u_{*S} = [\epsilon_{\text{opt}}(Z_H)\ K\ Z_H/h(\zeta)]^{1/3}\ .$$

The scintillation values of temperature flux are

$$-[T_* u_*]_S = \frac{-\zeta}{|\zeta|}\{Z_H^{2/3}\, C_T^2/G(\zeta)\}^{1/2} u_{*S}\ ,$$

here C_T^2 is the same as used above in the definition of V. Alternatively

$$-[T_* u_*]_S = -\zeta u_{*S}^3\ T/g\ K\ Z_H\ .$$

APPENDIX B

CATEGORIZING AND PARTITIONING A SUBPROGRAM'S FORMAL PARAMETERS

Many ways have been devised to make programs more readable. In FORTRAN one of the biggest problems I have encountered in reading a program is seeing a subroutine (or function) call containing a long string of arguments and not knowing which arguments are inputs and which are outputs. Several years ago, I devised a simple convention for alleviating this problem, and I have found it extremely useful.

I divide *all* formal arguments into the following three categories:

(1) INPUT – Arguments that transmit information into the subprogram but whose values remain unchanged on output.

(2) INPUT/OUTPUT – Arguments used to transmit information both into and out of the subprogram, plus any other arguments that do not fit into categories 1 or 3.

(3) OUTPUT – Arguments that transmit information out of the subprogram but no information into the subprogram; that is, their values on input are irrelevant.

Categories 1 and 3 are very restrictive, whereas category 2, in addition to including the normal input/output arguments, is a catchall. Although it might seem advantageous to split category 2, for simplicity (and another reason to be discussed later), the number of categories should be limited to three.

To distinguish between the three argument categories in writing and referencing subprograms, the arguments should be ordered according to category and the categories separated by spaces. The following rules will ensure an unambiguous and uniform use of spaces.

(1) INPUTS should be followed by a space.

(2) INPUT/OUTPUTS should be bracketed by spaces.

(3) OUTPUTS should be preceded by a space.

(4) Any double spaces should be merged into single spaces.

The use of these rules is illustrated below for all seven possible combinations of occurrence. When you have only INPUT and/or OUTPUT arguments, one space is used.

```
CALL SUB1(IN1,IN2, OUT1,OUT2)

CALL SUB2(IN1,IN2 )

CALL SUB3( OUT1,OUT2)
```

When you have INPUT/OUTPUT arguments (with or without INPUT and/or OUTPUT arguments), two spaces are used.

```
CALL SUB4(IN1,IN2, INOUT1,INOUT2, OUT1,OUT2)

CALL SUB5(IN1,IN2, INOUT1,INOUT2 )

CALL SUB6( INOUT1,INOUT2, OUT1,OUT2)

CALL SUB7( INOUT1,INOUT2 )
```

It is evident from the above examples that if there were a fourth category there would be no simple way to distinguish between the two middle categories in certain combinations, hence the need for limiting the number of categories to three.

In the classification of arguments, two situations need special attention. Consider an argument that transmits information into a subprogram but whose value is incidentally changed within the subprogram because it is "no longer needed." Although it might be tempting to classify this as an input (category 1) argument, this is very dangerous; others using the subroutine might indeed need the value of the argument again. My advice is to avoid creating such arguments, because to do so is simply bad programming.

The second type of argument needing special attention is the argument that passes no information in and no information out; it is simply used for work space. Although this type of argument is not common, it is occasionally useful. Such arguments should be placed in the input/output category. Fortunately, an argument of this type can be distinguished from "normal" input/output arguments by its name. Since these arguments carry no information, there is no need to name them for their information content. Consequently, you can standardize on a name such as WKSP or IWKSP (for WorK SPace or Integer WorK SPace).

Although the application of the convention is simple when using one's own subprograms, the question remains as to what to do when using "canned" subprograms. I have found that the sequence of arguments in many canned subprograms already conforms to the above principle; there seems to be a natural tendency to order arguments in this way. When the ordering exactly conforms, I simply insert the appropriate spaces. When the ordering does not *exactly* conform, I scrupulously avoid using any spaces, thus giving myself a danger signal when later reading the code.

I believe the uniform use of this convention will contribute significantly to the orderliness of programs and will remove probably the greatest frustration in interpreting someone else's program, or even you own program a few weeks after it is written.

APPENDIX C

TRIGGER AND CLOCK PULSE SYSTEM

The sonic anemometer contains its own clock, which produces an output from each channel every 1/25 second. It is, therefore, advantageous for the data acquisition system to be synchronized with the sonic anemometer so that the relative phases of the sonic system and the ADC system will not drift with respect to each other. To accomplish this we obviously need to send a synchronization signal from the sonic electronics to the ADC (analog-to-digital converter) board. The ADC board was, in my opinion, poorly designed in that it needed rather complicated synchronization signals. Figure C1 illustrates those synchronization signals.

We built a box that used as its input the top trace of Fig. C1 (the 25 Hz clock from the sonic); its output to the ADC was the two bottom traces. During the time between the first trigger and the second the system digitizes all N channels (in our case N = 16) 1024 times and puts the data into buffer A. Between the second and third trigger pulses, the system digitizes the data and puts them into buffer B. Also, during this time it stores away the A-buffer data into files Hxxx.DAT and Lxxx.DAT. When the third trigger pulse comes, the system again starts putting the digitized data into the A-buffer, while in the background, storing the data from the B-buffer. It will continue to flip-flop between these two buffers until 32 buffers full have been collected and the run ends.

The actual digitization of an individual channel is initiated by a clock pulse. Thus every 0.04 seconds a series of N (N = 16) pulses comes along and digitizes all the channels. The one exception to this is that the ADC card needs one extra clock pulse after each trigger pulse.

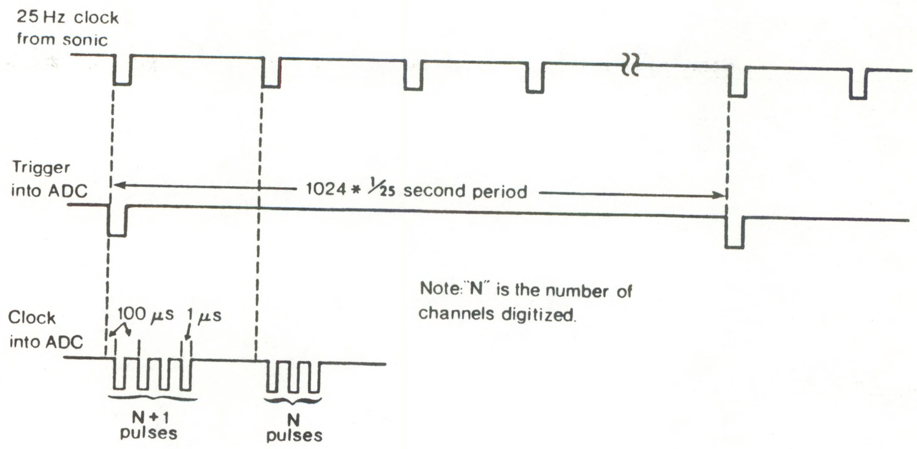

Fig. C1. The synchronization signals required by the ADC board and by the clock signal of the sonic anemometer.

APPENDIX D

CONVERSION TO ENGINEERING UNITS

The sixteen analog channels that are digitized and recorded by the system need to be converted to engineering units as a first step before the higher-level processing begins. This conversion takes place in subroutine ENGH_A or ENGH_B for the 4 high-frequency channels, and in subroutine ENGL_A or ENGL_B for the 12 low-frequency channels. All units are in SI unless otherwise indicated.

A concept that we use repeatedly is the counts per volt (CPV), since the data values are recorded in counts. All channels use the same ± 10 volt, 12–bit analog-to-digital converter (ADC), but the high- and low-frequency channels are treated differently in the recording process. Both high- and low-frequency channels are initially digitized at a 25 Hz rate. But, whereas the high-frequency channels are recorded at that rate, the low-frequency channels are each accumulated so that 16 of the 25 Hz samples are summed into one data point. Therefore the CPV for the high-frequency channels is

$$\mathrm{CPV}_{\mathrm{High}} = 2^{12}/20 = 204.8 \tag{D1}$$

and the CPV for the low frequency channels is

$$\mathrm{CPV}_{\mathrm{Low}} = (2^{12}/20) \times 16 = 3276.8\ . \tag{D2}$$

For each channel described in the appendix, the signal voltage from an instrument is amplified by a factor PAG (panel amplifier gain) and then digitized and recorded. Thus, from the recorded integer data IDAT the voltage from each of the high-frequency instruments is

$$V_{High} = IDAT/(204.8 \times PAG)\ , \tag{D3}$$

and that from the low-frequency instruments is

$$V_{\mathrm{Low}} = \mathrm{IDAT}/(3276.8 \times \mathrm{PAG})\ . \tag{D4}$$

The conversions used for the individual channels will now be discussed.

D1. High-Frequency Channels 1–3: Sonic Anemometer

Each of the three sonic anemometer axes operates similarly, thus I need only discuss the operation of one axis. The electronics for a given axis counts a 10 MHz oscillator for the time interval Δt corresponding to the time-of-flight difference between the sound propagating each way along the axis. For each sample output (sample outputs occur at a 25 Hz rate) eight such counts are taken, accumulated, and then displayed on the card-edge light emitting diodes (LEDs). Each such digital sample, less the two least significant bits, is also passed along to a 12-bit DAC (digital-to-analog converter). Thus if we designate a sample output or sample count as C, and the count that goes to the DAC as C_{DAC}, then we have the relation

$$C_{\mathrm{DAC}} = C/4 \; . \qquad \text{(D5)}$$

From the summation of the eight subsamples from the 10 MHz oscillator we have the relation

$$\Delta t = \frac{C}{8 \times 10^7} \quad \text{seconds} \; . \qquad \text{(D6)}$$

Therefore,

$$\Delta t = \frac{C_{\mathrm{DAC}}}{2 \times 10^7} \quad \text{seconds} \; . \qquad \text{(D7)}$$

The 12-bit DAC has a full-scale output of ±10 volts; thus the $\mathrm{CPV_{DAC}}$ (counts per volt for the DAC) is

$$\mathrm{CPV_{DAC}} = 2^{12}/20 = 204.8 \qquad \text{(D8)}$$

If the voltage out of the DAC is V_{DAC} then the count into the DAC (C_{DAC}) is

$$C_{\mathrm{DAC}} = \mathrm{CPV_{DAC}} \times V_{\mathrm{DAC}} \; . \qquad \text{(D9)}$$

Substituting this into Eq. (D7) gives

$$\Delta t = \frac{\mathrm{CPV}_{\mathrm{DAC}}}{2 \times 10^7} \times V_{\mathrm{DAC}}$$

$$= \frac{204.8}{2 \times 10^7} \times V_{\mathrm{DAC}} \qquad \text{(D10)}$$

Equation (D10) gives the time difference Δt when we know the analog output voltage from a sonic anemometer channel.

Since the voltage V_{DAC} in Eq. (D10) is the output of a high-frequency instrument, we may replace it with the value of V_{High} from Eq. (D3). Thus we have

$$\Delta t = \frac{204.8}{2 \times 10^7} \times \frac{\mathrm{IDAT}}{204.8 \times \mathrm{PAG}}$$

$$= \frac{5 \times 10^{-8}}{\mathrm{PAG}} \times \mathrm{IDAT}\ . \qquad \text{(D11)}$$

The velocity component Vel_d along a sonic axis of path length d meters is:

$$\mathrm{Vel}_d = \frac{201.5\overline{T}}{d} \Delta t \qquad \text{(D12)}$$

where $\overline{T}$ is the average absolute temperature (Kaimal, J.C., 1980). Substituting Eq. (D11) into Eq. (D12) gives

$$\mathrm{Vel}_d = \frac{201.5\overline{T}}{d} \; \frac{5 \times 10^{-8}}{\mathrm{PAG}} \times \mathrm{IDAT}\ . \qquad \text{(D13)}$$

D2. High-Frequency Channel 4: Fine-Wire T′ Sensor

A 1°C change in temperatuare causes a 1 volt change in the output of the T' sensor. Thus, using Eq. (D3), we find the temperature change ΔT to be

$$\Delta T = \frac{1}{204.8 \times \mathrm{PAG}} \times \mathrm{IDAT}\ . \qquad \text{(D14)}$$

D3. Low-Frequency Channels 1–2: Large-Aperture Scintillometers

The large-aperture scintillometers preprocess the scintillation signals so that

$$C_n^2 = C D_L^{7/3} L^{-3} \sigma_\chi^2 \tag{D15}$$

where $C = 4.48$, $D_L = 0.146$ m, $L = 607$ m, and σ_χ^2 is the log-amplitude variance. σ_χ^2 is defined as one-fourth of the variance of the square of the signal from the photodetector. The signal bandwidth between half-power points is 0.05–1300 Hz, and the output time constant $T = 1$ second. The outputs are logarithmic and scaled so that the refractive-index structure parameter in (m$^{-2/3}$) is

$$C_n^2(Z_L) = 10^{(V_1-14)} \tag{D16}$$

$$C_n^2(Z_H) = 10^{(V_2-14)} \tag{D17}$$

where V_1 or V_2 is the voltage from a low-frequency instrument and thus may be replaced by V_{Low} from Eq. (D4).

D4. Low-Frequency Channels 3–4: Inner-Scale Meter

The inner-scale meter preprocesses the scintillation signal from the coherent light path such that

$$C_{n,\text{Laser}}^2 = \frac{\sigma_\chi^2}{0.124 k^{7/6} L^{11/6}} \tag{D18}$$

where $k = 2\pi/(0.6328 \times 10^6)$m, $L = 150$ m, and σ_χ^2 is the log-amplitude variance of the laser light at the detector. The signal bandwidth is 0.1–5000 Hz, and the output time constant is 1 second.

The incoherent light signal is preprocessed such that

$$C_{n,\text{Laser}}^2 = C D_{\text{LA}}^{7/3} L^{-3} \sigma_{\text{LA}}^2 \tag{D19}$$

where $C = 4.48$, $D_{\mathrm{LA}} = .044$ m, and $L = 150$ m, and σ_{LA} is the log-amplitude variance. The signal bandwidth is 0.04–2600 Hz and the output time constant is 1 second.

The outputs of the inner-scale meter are also logarithmic, so

$$C^2_{n,\mathrm{Laser}} = 10^{(V_3-14)} \tag{D20}$$

$$C^2_{n,\mathrm{Laser}} = 10^{(V_4-14)} \tag{D21}$$

where again the voltages V_3 and V_4 may be replaced by V_{Low} from Eq. (D4).

D5. Low-Frequency Channel 5: WOPT

The vertical wind measurement system employs a 15–cm–diameter LED transmitter and two vertically-spaced tangent receiving apertures, also 15 cm in diameter. The slope of the covariance of the received signals is measured at zero time delay, and the slope-wind calibration used is that derived for horizontal winds and aperture separation. The calibration is such that

$$W_{\mathrm{opt}} = -\,1.67 \times V_5 \tag{D22}$$

in meters per second; an upward wind produces a positive value for W_{opt}. (The FORTRAN variable is WOPT.) The output time constant is 10 seconds.

D6. Low-Frequency Channel 6: Wind Structure Function

The wind structure function is measured with two vertically spaced, platinum-film-on-quartz, hot-film probes operated in the constant temperature mode. These data are preprocessed by first taking the difference of the two signals, then filtering the signal in the bandpass 0.02–10000 Hz and measuring the rms of this signal with a 10 second time constant. The calibration is derived as follows. By definition, the velocity structure parameter is

$$C^2_V = \frac{\overline{(v_a - v_b)^2}}{R_v^{2/3}} \tag{D23}$$

where v_a and v_b are the wind speeds at two points separated by the distance $R_v = 0.1$ m. The unit is calibrated so that

$$\sqrt{\overline{(v_a - v_b)^2}} = 9.144 \times V_6 \qquad \text{(D24)}$$

where V_6 is the output voltage. Thus

$$C_V^2 = \frac{9.144^2 \times V_6^2}{0.1^{2/3}}$$

$$= 388.1 \times V_6^2 \qquad \text{(D25)}$$

This is the conversion equation used in versions 1.3 and earlier of the FLUX1 program.

We noticed, however, that the energy dissipation rate implied by C_V^2 and that implied by the optical inner-scale results were in rather poor agreement. In searching for possible causes for this poor agreement, we found that the rms converter output had an offset voltage of 6.8 mV with its input shorted. This, in fact, could cause significant error when the input signal was small, as it was in several cases.

To mitigate the error caused by the rms converter, we installed a precision ×10 amplifier to boost the input signal to the converter, and we decided to also compensate for the error, as best we could, with software modifications. The software modification would, of course, be different for the runs made prior to the ×10 amplifier installation (runs 1–16) and those after the installation (runs 17 and higher).

This voltage offset, depending upon where it occurred in the converter circuit, might add either linearly ($V_{out} = V_{in} + V_{offset}$) or orthogonally ($V_{out} = (V_{in}^2 + V_{offset}^2)^{1/2}$) to the input voltage. To determine what model to use we fed a 100 Hz signal into the input of the converter and measured its dc output voltage as a function of the rms input. Table D1 gives the results of this measurement along with three models of it:

(1) The linear model:

$$V_{out} = V_{in} + 0.0068 \; . \qquad \text{(D26)}$$

(2) The orthogonal model:

$$V_{out} = (V_{in} + 0.0068^2)^{1/2} \; . \qquad \text{(D27)}$$

(3) A mixed model:

$$V_{out} = (V_{in}^2 + 0.0053^2)^{1/2} + 0.0015 \; . \qquad \text{(D28)}$$

Table D1. Comparison between the measured output of the rms-to-dc converter and three attempts to model its output, including percentage errors for each model

Measured		Linear model		Orthogonal model		Mixed model	
V_{in} (mV)	V_{out} (mV)	V_{out} (mV)	Percent error	V_{out} (mV)	Percent error	V_{out} (mV)	Percent error
0.0	6.8	6.8	0.00	6.80	−0.00	6.80	0.00
1.0	6.9	7.8	+13.04	6.87	−0.43	6.89	−0.14
2.0	7.15	8.8	+23.08	7.09	−0.84	7.16	+0.14
3.0	7.6	9.8	+28.95	7.43	−2.24	7.59	−0.13
4.0	8.2	10.8	+31.71	7.89	−3.78	8.14	−0.73
6.0	9.6	12.8	+33.33	9.07	−5.52	9.51	−0.94
8.0	11.2	14.8	+32.14	10.50	−6.25	11.10	−0.89
10.8	12.9	16.8	+30.23	12.09	−6.28	12.89	−0.08
12.0	14.8	18.8	+27.03	13.79	−7.32	14.62	−1.22
14.0	16.6	20.8	+25.30	15.56	−6.27	16.47	−0.78
20.0	22.3	26.8	+20.18	21.12	−5.29	22.19	−0.49
30.0	32.0	36.8	+15.00	30.76	−3.87	31.96	−0.12
40.0	41.9	46.8	+11.69	40.57	−3.17	41.85	−0.12
49.9	51.36	56.7	+9.67	50.36	−2.59	51.68	−0.04
60.0	61.7	66.8	+8.27	60.38	−2.14	61.73	−0.05
79.9	81.25	86.7	+6.83	80.19	−1.30	81.58	+0.41
100.0	101.2	106.8	+5.53	100.23	−0.96	101.64	+0.43

Along with each model we give the percentage error of that model, where

$$\text{percentage error} = \frac{V_{\text{out-model}} - V_{\text{out-measured}}}{V_{\text{out-measured}}} \times 100\% \ . \qquad \text{(D29)}$$

The table indicates that the mixed model matches the rms converter quite accurately.

Upon inverting the mixed model equation we have,

$$V_{in} = [(V_{out} - .0015)^2 - .0053]^{1/2} \ . \qquad \text{(D30)}$$

Substituting this into Eq. (D25) gives

$$C_V^2 = 388.1 \times ((V_{out} - .0015)^2 - .0053) \ . \qquad \text{(D31)}$$

This equation, used in subroutine ENGL_A of program FLUX1, should substantially improve the accuracy of C_V^2 for those runs (1–16) prior to the installation of the $\times 10$ amplifier. Subroutine ENGL_B, used for the runs (17–52) after the amplifier installation, uses the relation

$$C_V^2 = 3.881 \times ((V_{\text{out}} - .0015)^2 - .0053) \ , \tag{D32}$$

which reflects the gain of the precision amplifier, as well as the correction for the imperfect rms converter, although that correction is far less important with the amplifier.

Version 1.8 of program FLUX1 senses the run number and uses the appropriate conversion equation, thus it is good for all run numbers.

D7. Low-Frequency Channels 7–8: Temperature Structure Parameters

The temperature structure parameters CT2LO and CT2HI are each measured with vertically spaced, fast-response platinum temperature probes 2.5 μm in diameter. The signals are preprocessed by taking the difference, then filtering through a bandpass of 0.02–500 Hz (the high frequency determined by the wire response), and then calculating the logarithm of the mean square, averaged over 1 second. The instrument calibration is checked by generating internally square waves having peak-to-peak values equivalent to 0.1° and 1.0°C, based upon a temperature coefficient of resistance for platinum of 0.00386. By definition,

$$C_T^2 = \frac{\overline{(T_a - T_b)^2}}{R_T^{2/3}} \ , \tag{D33}$$

where R_T is the separation of the temperature sensors in meters. The calibration is set up so that

$$C_T^2 = \frac{10^{(V-1.6)}}{R_T^{2/3}} \ . \tag{D34}$$

For R_T = .1 m,

$$C_T^2(Z_L) = 4.64 \times 10^{(V_7-1.6)} \tag{D35}$$

and

$$C_T^2(Z_H) = 4.64 \times 10^{(V_8-1.6)} \ . \tag{D36}$$

D8. Low-Frequency Channels 9–10: Propeller Vane Anemometer

The propeller vane anemometer (propvane) is an R.M. Young model 8002 instrument. The propeller-generator with the associated power supply-translator is calibrated to an output of 100 mV per meters per second of wind speed. Therefore if V_9 is the output voltage from the propeller system our conversion equation is

$$\mathrm{PROP} = 10. \times V_9 \ . \qquad \text{(D37)}$$

The vane with the associated power supply-translator has an output of 10 mV per azimuth degree with increasing voltage as the vane rotates clockwise as viewed from above. The propvane was mounted on the rotatable boom, and thus produced an output relative to the boom. We desired that the vane output be zero for wind blowing directly into the boom and to increase as the vane rotated counterclockwise as viewed from above. (This sense of rotation was needed to match the convention previously established for the sonic anemometer.) Also, the angle-sensing potentiometer had a dead spot between 352° and 360°, so we rotated the index (zero-output point) of the vane 180° from the front of the boom. This gives a conversion equation of

$$\mathrm{VAN} = 180 - (100 \times V_{10}) \qquad \text{(D38)}$$

where VAN is the azimuth in degrees and V_{10} is the vane voltage output. For our use, however, we desired the azimuth in radians. Therefore we used the equation

$$\mathrm{VAN} = (\pi/180) \times [180 - (100 \times V_{10})] = 3.1415926 - 1.745329 \times V_{10} \ . \qquad \text{(D39)}$$

D9. Low-Frequency Channel 11: Barometer

The atmospheric pressure was measured by the Rosemount pressure transducer (model number 1201F1A3A1A, and serial number 2382) that was originally purchased for the WPL millimeter-wave experiment in Flatville, Illinois. The original conversion equation was

$$P = 800 + (60 \times V_{11}) \qquad \text{(D40)}$$

where P is in millibars and V_{11} is output volts.

Atmospheric Instrumentation Research, Inc. ran a new calibration on the instrument. The result is shown in Table D2. These data indicate that the instrument is still quite linear but that the sensitivity has changed moderately. Using the endpoints to determine the sensitivity and the region near 800 mbar to determine the offset, I arrived at the new calibration equation:

$$P = 800.04 + (62.06 \times V_{11}) \ . \tag{D41}$$

Table D2. Calibration of Rosemount pressure transducer

Pressure (mbar)	Voltage (V)
600.13	−3.2340
650.37	−2.4173
700.03	−1.6124
750.21	−0.8015
800.02	−0.0003
849.86	+0.8007
875.05	1.2054
899.94	1.6050
950.07	2.4095
1000.29	3.2142

D10. Low-Frequency Channel 12: Boom Angle

A precision angle encoder was used to monitor the boom angle ANGL. The encoder was arranged so that its output V_{12} was 0 volts with the boom facing east, and increased to 10 volts as the boom rotated clockwise (as viewed from above) through one revolution. Thus the azimuth angle in radians was

$$\mathrm{ANGL} = 1.5\pi + (2\pi/10) \times V_{12}$$

for the quadrant from west to north, and

$$\mathrm{ANGL} = 1.5\pi + [(2\pi/10) \times V_{12}] - 2\pi$$

for the three quadrants from north to east to south to west. These relations can be combined into one FORTRAN statement using the AMOD function.

$$\mathrm{ANGL} = \mathrm{AMOD}[(1.5 * \mathrm{PI} + (2 * \mathrm{PI}/10) * V_{12}),\ 2 * \mathrm{PI}] \ . \tag{D42}$$

APPENDIX E

EXAMPLE OF A PARAMETER FILE

FILE: P025.DAT

```
´TCBAR   ´    13.1       / average temp (deg. C)
´ZH      ´     3.95      / High instrument height (m)
´ZL      ´     1.45      / Low instrument height (m)
´RHBAR   ´    35.0       / average Relative Humidity (%)
´NHICH   ´      4        / no. of High freq CHannels
´NLOCH   ´     12        / no. of LOw freq CHannels
´NHPTT   ´    32768      / No. of HIgh freq data PoinTs per ch. for run
´NLPTT   ´     2048      / No. of Low freq data PoinTs per ch. for run
´IDATBEG´ 1988 10 17  / iyr, imo, iday
´ITIMBEG´ 15 28 00 00 / ihr, imin, isec, i100th
´IDATEND´ 1988 10 17  / iyr, imo, iday
´ITIMEND´ 15 49 50 72 / ihr, imin, isec, i100th
*
*
*
CH#        PAG        / Channel # and Panel Amplifer Gain
 1          1
 2          1
 3          1
 4          2
 5          2
 6          2
 7          2
 8          2
 9          2
10          2
11          2
12          2
13          1
14          1
15          1
16          1
```

APPENDIX F

EXAMPLE OF AN OUTPUT FILE

FILE: F025.V19

```
                 THE 1988 MICROMETEOROLOGICAL
                   SCINTILLATION EXPERIMENT

Run:   025                      Beginning   1988-10-17  15:28:00.00

Temp:       13.10 deg C         FLUX1  ver. 1.9
Press:     833.34 mb
RH:         35.00 %
ZH:          3.95 m
ZL:          1.45 m
Boom ang:   75.95 deg

1. PROCESSING OF SONIC ANEMOMETER AND T´ DATA

We define three measurments of the time-averaged wind
direction (relative to the boom) as:
   <theta(1)> = ATAN(<v>/<u>)            from sonic
   <theta(2)> = ATAN(-<v´w´>/-<u´w´>)    from sonic
   <theta(3)> = ATAN(<v>/<u>)            from prop-vane

              wind angle     vertical flux of horizontal
              relative to    momentum per unit mass-density
             | ------------ | ------------------------------|
             | boom  | north | along-stream  | cross-stream  |
             |-------|-------|---------------|---------------|
theta(1)     | -10.40|  86.35|   -0.05741    |    0.01103    |
theta(2)     | -21.28|  97.22|   -0.05846    |    0.00000    |
theta(3)     |  -8.53|  84.48|   -0.05702    |    0.01290    |

              friction vel.  temp scaling    Monin-Obukhov
                              parameter          length
             |---------------|---------------|---------------|
theta(1)     |    0.23961    |   -0.18858    |   -22.23131   |
theta(2)     |    0.24179    |   -0.18688    |   -22.84346   |
theta(3)     |    0.23879    |   -0.18923    |   -22.00453   |

Note: The wind angle relative to the boom is 0 degrees for
wind blowing into the face of the boom, and increases as the
wind vector moves counter clockwise--as viewed from above.
The boom angle is 0 degrees when the boom is facing north and
increases as the boom rotates clockwise--as viewed from above.
Thus the geographic wind angle (relative to north) is the
boom angle minus the boom-relative wind angle.
```

```
                SONIC AND T′ QUANTITIES

                  Quarterly Averages          Final Ave
          ---------------------------------- ---------
             1         2         3        4
U         |  3.1273 |  3.2258 |  2.8852 |  2.6475 |  2.9714 |
V         | -0.8219 | -0.9571 |  0.2529 | -0.6563 | -0.5456 |
W         |  0.1515 |  0.1374 |  0.1733 |  0.1437 |  0.1515 |
T         |  0.4708 |  0.5054 |  0.4499 |  0.2962 |  0.4306 |
UW        |  0.4248 |  0.3853 |  0.4411 |  0.3312 |  0.3956 |
VW        | -0.0887 | -0.0968 |  0.0298 | -0.0900 | -0.0614 |
WW        |  0.1389 |  0.1284 |  0.1327 |  0.1029 |  0.1257 |
TW        |  0.1330 |  0.1065 |  0.1277 |  0.0743 |  0.1104 |
TT        |  0.4009 |  0.3560 |  0.3822 |  0.1468 |  0.3215 |
speed     |  3.2971 |  3.4966 |  3.0084 |  2.8531 |  3.1638 |
temp flux|  0.0617 |  0.0371 |  0.0498 |  0.0318 |  0.0452 |
theta(1)  | -14.724 | -16.526 |   5.010 | -13.923 | -10.404 |
theta(2)  | -36.220 | -30.880 |  13.411 |  -5.024 | -21.275 |

                  PROP-VANE QUANTITIES

U         |  3.5406 |  3.6076 |  3.2729 |  3.0898 |  3.3777 |
V         | -0.7876 | -0.8826 |  0.2569 | -0.6134 | -0.5067 |
speed     |  3.6729 |  3.8177 |  3.3607 |  3.2434 |  3.5237 |
theta(3)  | -12.542 | -13.747 |   4.488 | -11.228 |  -8.531 |

                   ′Kilo′ Histo-table of
              Sonic Wind Direction (15-degree bins)

  Q   || --  -6  -5  -4| -3  -2  -1   1   2   3 | 4   5   6  ++|
  1   ||  0|  0|  0|  4| 93|380|430| 91|  2|  0|  0|  0|  0|  0|
  2   ||  0|  0|  0| 15|184|302|292|172| 33|  0|  0|  0|  0|  0|
  3   ||  0|  0|  0|  0| 16|121|205|381|230| 47|  0|  0|  0|  0|
  4   ||  0|  0|  1| 23|106|363|258|177| 69|  3|  0|  0|  0|  0|
Final||  0|  0|  0| 11|100|291|296|205| 84| 13|  0|  0|  0|  0|

                   ′Kilo′ Histo-table of
            Prop-vane Wind Direction (15-degree bins)

  Q   || --  -6  -5  -4| -3  -2  -1   1   2   3 | 4   5   6  ++|
  1   ||  0|  0|  0|  0| 33|352|543| 72|  0|  0|  0|  0|  0|  0|
  2   ||  0|  0|  0|  2|127|307|365|186| 14|  0|  0|  0|  0|  0|
  3   ||  0|  0|  0|  0|  0| 84|270|443|193| 10|  0|  0|  0|  0|
  4   ||  0|  0|  0|  0| 74|365|322|205| 33|  0|  0|  0|  0|  0|
Final||  0|  0|  0|  0| 59|277|375|227| 60|  2|  0|  0|  0|  0|
```

2. INNER SCALE FROM SCINTILLATION AND CV2

INSTRUMENT OUTPUTS

	Quarterly Averages				Final Ave
	1	2	3	4	
CN2LO	1.540E-13	1.051E-13	6.706E-14	5.132E-14	9.438E-14
CN2HI	6.115E-14	4.842E-14	3.460E-14	2.468E-14	4.221E-14
CN2LSR	2.562E-14	1.843E-14	1.203E-14	8.185E-15	1.607E-14
CN2LA	7.694E-14	5.645E-14	3.843E-14	3.353E-14	5.134E-14
Wopt	0.0327	0.2040	0.2291	0.0582	0.1310
CV2	0.2662	0.2649	0.2170	0.1962	0.2361
CT2LO	0.2004	0.1282	0.0818	0.0616	0.1180
CT2HI	0.0751	0.0448	0.0383	0.0247	0.0457
BARM	833.3229	833.3405	833.3405	833.3427	833.3367
BM_ANG	75.9497	75.9491	75.9486	75.9481	75.9489

```
y     =   CN2LSR / CN2LA         =           0.31293
l_opt   (from y and TABLE 1)     =           0.00684

CN2_Laser_RI   (from Rytov variance with Inner scale) =  1.607E-14

y_RI   =   CN2_Laser_RI / CN2LA          =        0.31293
l_opt_RI   (from TABLE 1 using y_RI)     =        0.00684
```

Note: We will use the name SLASR to indicate the TABLE 1 correction factor derived from l_opt, and SLASR_RI to indicate the similar factor derived from l_opt_RI. Similarly for SLAP.

```
CN2_LA_RI_IS      =  CN2_LA / SLAP_RI              =        4.032E-14
CN2_Laser_IS      =  CN2_Laser / SLASR             =        4.099E-14
CN2_Laser_RI_IS   =  CN2_Laser_RI / SLASR_RI       =        4.099E-14

epsilon_opt_RI   =  0.7413E-02
epsilon_v        =  0.2634E-01

l_v              =    0.004981

CN2_IS_RI_(ZH)   =  CN2_(ZH) / SLAPL_RI   =    4.133E-14
CN2_IS_V_(ZH)    =  CN2_(ZH) / SLAPL_V    =    4.192E-14
```

CN2 Ratios at Z = 3.95 meters

```
CN2_Laser_IS     / CN2_LA_RI_IS   =    1.017
CN2_Laser_RI_IS  / CN2_LA_RI_IS   =    1.017
CN2_IS_RI_(ZH)   / CN2_LA_RI_IS   =    1.025
CN2_IS_V_(ZH)    / CN2_LA_RI_IS   =    1.040
```

The inner scale is now corrected to the lower height.
(The subscripts "1" and "2" indicate use of the M-O length derived from theta_1 and theta_2 respectively.)

```
l_L_opt_RI_1  =   0.00545
l_L_v_1       =   0.00397
l_L_opt_RI_2  =   0.00545
l_L_v_2       =   0.00397
```

The Non-Dimensional Energy Dissipation Rate NDEDR has subscripts "opt" or "v" indicating the instrument (optical or thin-film differential velocity) and 1 or 2 indicating the mean-wind-direction computational method (theta_1 or theta_2).

```
NDEDR_opt_1    =      0.85140
NDEDR_v_1      =      3.02495
ZH/MON_OBK 1   =     -0.17768

NDEDR_opt_2    =      0.82859
NDEDR_v_2      =      2.94389
ZH/MON_OBK 2   =     -0.17292
```

3. COMPARISON OF LARGE-APERTURE CN2 VALUES WITH SPACED-WIRE CT2

```
CT2HI / CT2_OPT_HI   =    0.6817
CT2LO / CT2_OPT_LO   =    0.7868

                            theta_1       theta_2       theta_3
                          ----------    ----------    ----------
NDIM_CT2_HI          =        3.2129        3.2716        3.1910
NDIM_CT2_OPT_HI      =        4.7130        4.7992        4.6809
ZH / L_MO            =       -0.1777       -0.1729       -0.1795

NDIM_CT2_LO          =        4.2509        4.3285        4.2219
NDIM_CT2_OPT_LO      =        5.4028        5.5015        5.3660
ZL / L_MO            =       -0.0652       -0.0635       -0.0659
```

```
4. STABILITY FROM TWO-HEIGHT CN2 AND CT2

R_n         =      0.87233
R_T         =      0.75582
ZETA_n      =     -0.03582
ZETA_T      =     -0.10234
L_MO_n      =    -66.81247
L_MO_T      =    -23.38492

              STABILITY PARAMETERS USING SECTION-1 L_MO

                  theta_1          theta_2          theta_3
                -----------      -----------      -----------
L_MO    =       -22.23131        -22.84346        -22.00453
ZETA    =        -0.10765         -0.10477         -0.10876

A_n     =         5.93701          6.04550          5.89657
B_n     =         2.32919          2.39333          2.30543
C_n     =     ???????????      ???????????      ???????????
D_n     =     ???????????      ???????????      ???????????

A_T     =         5.40568          5.50446          5.36886
B_T     =         6.65468          6.83792          6.58680
C_T     =     ???????????      ???????????      ???????????
D_T     =     ???????????      ???????????      ???????????

5. THE MEANING OF OPTICALLY-DERIVED W

                 ZETA       Wopt/frict_vel
          |------------|----------------|
theta(1)  |   -0.1777  |      0.5467    |
theta(2)  |   -0.1729  |      0.5417    |
theta(3)  |   -0.1795  |      0.5485    |

6. THE MONIN-OBUKHOV SIMILARITY OF <T′**2> AND <W′**2>

                 ZETA       <T′**2>/TEMP_SP**2      <w′**2>/frict_vel**2
          |------------|----------------------|----------------------|
theta(1)  |   -0.1777  |        3.82727       |        1.79067       |
theta(2)  |   -0.1729  |        3.89720       |        1.75853       |
theta(3)  |   -0.1795  |        3.80119       |        1.80295       |
7. FLUXES FROM PROPAGATION

V =      0.17612

               Parameters for the Unstable Atmosphere

              zeta         M-O        friction    temp flux    temp flux
                          length      velocity        A            B
          |-----------|-----------|-----------|-----------|-----------|
Wyngaard  |  -0.35294 | -11.19156 |   0.20479 |   0.05604 |   0.05604 |
Mod. Wyn. |  -0.32300 | -12.22920 |   0.20590 |   0.05214 |   0.05212 |
Kohsiek   |  -0.30370 | -13.00614 |   0.20664 |   0.04942 |   0.04954 |

                Parameters for the Stable Atmosphere

              zeta         M-O        friction    temp flux    temp flux
                          length      velocity        A            B
          |-----------|-----------|-----------|-----------|-----------|
Wyngaard  |   0.26324 |  15.00522 |   0.15929 |  -0.01973 |  -0.01967 |
Mod. Wyn. |   0.29984 |  13.17384 |   0.15610 |  -0.02112 |  -0.02108 |
```

APPENDIX G

SOURCE CODE FOR THE REAL-TIME TABULAR DATA DISPLAY MODULE

All of the C source files in this appendix were compiled under the Microsoft C compiler version 5.1 using the compiler command CL/GT /AL filename.C. The FORTRAN source files were compiled under the Ryan-McFarland FORTRAN compiler version 2.43 using the command RMFORT filename /LIXY.

FILE: DISP.BAT

```
COPY DISP_CH.BAT D:
D:
DISP_CH
```

FILE: DISP_CH.BAT

```
COPY C:\FLUX\EXE\CC.EXE
COPY C:\FLUX\EXE\READP.EXE
COPY C:\FLUX\EXE\DISP_CH1.EXE
READP
ECHO OFF
CLS
:MARK
CC FIDAT
DISP_CH1
GOTO MARK
```

FILE: CC.C

```
#include <vp.h>
main(argc,argv)
int argc;
char *argv[];
{
int nch;
int outf;
outf=open(argv[1],O_BINARY|O_WRONLY|O_CREAT,S_IWRITE);
if(argc != 2)
  {
  puts("cc outfile");
  exit(1);
  }
Clock(20000.);
reset();
setAD(0,15,16);
ComReady();
SetDMA(3,0,16);
Dgtzich();
write(outf,0x30000000,32);
}
```

FILE: VP.H

```
#include<math.h>
#include<stdlib.h>
#include<stdio.h>
#include<io.h>
#include<memory.h>
#include<dos.h>
#include<fcntl.h>
#include<sys\types.h>
#include<sys\stat.h>
#include<string.h>
```

FILE: DTLIB.C

(This file is too long to print here. It is stored on the enclosed diskette in directory AP-PNDX_G)

FILE: READP.FOR

```
      PROGRAM READP
C
C     Programmed by JTP Aug 30,1988.
C
C     This program reads the file C:\FLUX\P000.DAT, gets the array PAG
C     of Panel Amplifier Gains and writes it into file D:FPAG.
C
      REAL PAG(16)
      CHARACTER CH7*7
C
      OPEN(10,STATUS='OLD',FILE='C:\FLUX\P000.DAT',ACCESS='SEQUENTIAL',
     *     FORM='FORMATTED')
C
      DO 10 I=1,20
         READ(10,'(A7)') CH7
         IF(CH7.EQ.'CH#    ') GOTO 20
10    CONTINUE
20    CONTINUE
C
      DO 30 I=1,16
         READ(10,*) K,PAG(I)
         IF(K.NE.I) THEN
            WRITE(*,101) I
101         FORMAT(' prog READP: out of sequence at I = ',I2)
```

```
            STOP
         ENDIF
30    CONTINUE
      CLOSE(10)
C
      OPEN(11,STATUS='UNKNOWN',FILE='FPAG',ACCESS='DIRECT',
     *     FORM='UNFORMATTED',RECL=64)
      WRITE(11,REC=1) PAG
      CLOSE(11)
C
      STOP
      END
```

FILE: DISP_CH1.FOR

```
      PROGRAM DISP_CH1
C
C     Programmed by JTP Aug 30,1988.
C
C     This program reads files FPAG and FIDAT from D: and displays the
C     instrument voltages and engineering units on the screen. It is to
C     be used from READ_CH.BAT.
C
      REAL PAG(16), V1(16), V2(16), E1(16)
      INTEGER*2 IDAT(16)
      CHARACTER*7 NAMES(16)
      CHARACTER ESC*1
C
      OPEN(10,STATUS='OLD',FILE='FPAG',ACCESS='DIRECT',
     *     FORM='UNFORMATTED',RECL=64)
      READ(10,REC=1) (PAG(I),I=1,16)
      CLOSE(10)
C
      OPEN(11,STATUS='OLD',FILE='FIDAT',ACCESS='DIRECT',
     *     FORM='UNFORMATTED',RECL=32)
      READ(11,REC=1) (IDAT(I),I=1,16)
      CLOSE(11)
C
      DATA NAMES/'sonic A','sonic B','sonic W','T'     ','CN2LO  ',
     *           'CN2HI  ','CN2LSR ','CN2LA  ','WOPT   ','CV2    ',
     *           'CT2LO  ','CT2HI  ','PROP   ','VAN    ','BARM   ',
     *           'ANGL   '/
C
C     Convert IDAT from offset binary to 2's complement then compute V2
C     and V1.
C
      COEFV2 = 1.0/204.8
      DO 10 I=1,16
         IF(IDAT(I).GE.2048) THEN
            IDAT(I) = IDAT(I) - 2048
         ELSE
            IDAT(I) = IDAT(I) + 63488
         ENDIF
         V2(I) = COEFV2*IDAT(I)
         V1(I) = V2(I)/PAG(I)
10    CONTINUE
C
      TC     = 20.0
      TK     = 273.15 + TC
      SEP    = .2
      SONIC = 201.5 * 5.0E-8 * TK * 204.8 / SEP
C
      E1(1)  = SONIC * V1(1)
      E1(2)  = SONIC * V1(2)
      E1(3)  = SONIC * V1(3)
      E1(4)  = V1(4)
      E1(5)  = 10.**(V1(5)-14.)
      E1(6)  = 10.**(V1(6)-14.)
      E1(7)  = 10.**(V1(7)-14.)
      E1(8)  = 10.**(V1(8)-14.)
      E1(9)  = -1.66666666 * V1(9)
      E1(10) = 15.52 * V1(10)**2
      E1(11) = 4.641589 * 10.**(V1(11)-1.6)
      E1(12) = 4.641589 * 10.**(V1(12)-1.6)
      E1(13) = 10. * V1(13)
      E1(14) = 100. * V1(14) - 180.
      E1(15) = 800. + 60. * V1(15)
      E1(16) = AMOD((270.+36.*V1(16)),360.)
C     E1(16) assumes that the angle encoder is zeroed to the west.
C
      ESC = CHAR(27)
      WRITE(*,101) ESC
101   FORMAT(1X,A1,'[1;1H',1X,20X,'INSTRUMENT OUTPUTS')
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,102)
102   FORMAT(1X,'CH#   instrument   inst volts   PAG    amp volts',
     *       '      eng units ')
      WRITE(*,*)
C
      DO 20 I=1,16
         WRITE(*,103) I, NAMES(I), V1(I), PAG(I), V2(I), E1(I)
103      FORMAT(1X,I2,1X,3X,A7,3X,3X,F10.3,3X,1X,F2.0,3X,F10.3,4X,
     *          1PG11.3)
20    CONTINUE
C
C     DO 30 I=1,10000
C        AA = SIN(FLOAT(I)*.001)
30    CONTINUE
```

```
STOP
END
```

APPENDIX H

SOURCE CODE FOR THE DATA ACQUISITION MODULE

FILE: FLUXD.BAT

```
DEL D:H???.DAT
FLUXDAT L%1.DAT D:H%1.DAT 16 32 4
```

FILE: GETDAT.BAT

```
GETHFF %1 A /R2048
COPY L%1.DAT A:
```

FILE: FLUXDAT.C

```
/*************************************************************************
         Program Fluxdat.c digitizes nconv samples per channel
         for nblocks. Program will be used for data collection
         at Table Mountain field site for the Flux Experiment
         August-September 1988.
         I really appreciate the help of Jim Churnside, Mike Price,
         Mark Wickers, and Cindy Schreiber on this seemingly
         never ending project. THANKS!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

 to C&L: cl /c /WO /AH fluxdat.c <ret>
                 link fluxdat; <ret>
 NOTE: ANY LIBRARY ROUTINES NORMALLY FOUND IN DTLIB.C
          ARE INCLUDED AT THE END OF THE MAIN SOURCE CODE
          FOR LINKING CONVENIENCE.

 Written by Ray Harrison, July 1988.
*************************************************************************/

          .
          .
          .
```

(The complete FLUXDAT.C file is stored on the enclosed diskette in directory APPNDX_H.) The source files in this appendix used the same compilers and options as those in Appendix G.

FILE: GETHFF.FOR

```
      PROGRAM GETHFF
C
C     Programmed by JTP Aug.2-3,1988.
C
C     This program transfers the high freq file Hxxx.DAT on the D drive
C     (RAM disk) to the A, B, or C drive.  In the process of
C     transferring, two transformations (corrections) are performed.
C
C     The file on the D drive is composed of 1024 data points from
C     channel 1, 1024 from channel 2, etc. through channel 4.  THis
C     pattern of 1024 points from each of the 4 channels is repeated 32
C     times for a total of the 1024*4*32 = 131072 data points
C     (262144 bytes) in the file.  The first transformation is to
C     'de-interleave' the data so that all of the (1024*32 = 32768)
C     channel 1 data points come first, followed by all of the channel
C     2 data points and so forth.
C
C     The next transformation (or correction) corrects for a problem in
C     the data acquisition system.  This problem causes every 1024'th
C     data point in each channel to be a zero instead of the correct
C     data value.  Thus each data channel has 1023 correct data points
C     followed by a zero, followed by another 1023 correct points and a
C     zero, etc.  This program 'corrects' for this by doing a linear
C     interpolation for the 31 interior points of each channel and then
C     duplicating the last data point.
C
C     The calling sequence for this program is:
C
C        GETHFF  run_no  [A|B|C]  /R2048
C
C     where 'run_no' is an up-to-3-digit run number, [A|B|C] is either
C     an A, B, or C indicating the output disk drive, and the /R2048 is
C     simply necessary to allow 2048-byte I/O transfers.
C
      PARAMETER (NPTB=1024)
      INTEGER*2 IRUN, NCH, NBLKS, LENR, IN, IOUT
      INTEGER*2 IDAT(NPTB,2)
      CHARACTER DRIVE*2, IRUNCH*3, INPATH*10, OUTPATH*10
C
      NCH   = 4
      NBLKS = 32
      LENR  = 2*NPTB
      IN    = 10
      IOUT  = 11
C
      CALL CHKCMD( IRUN,IRUNCH,DRIVE )
      WRITE(INPATH, '(2HD:,1HH,A3,4H.DAT)') IRUNCH
      WRITE(OUTPATH,'(A2,1HH,A3,4H.DAT)') DRIVE, IRUNCH
C
```

```
      OPEN(UNIT=IN,STATUS='OLD',FILE=INPATH,ACCESS='DIRECT',
     *     FORM='UNFORMATTED',RECL=LENR)
      OPEN(UNIT=IOUT,STATUS='NEW',FILE=OUTPATH,ACCESS='DIRECT',
     *     FORM='UNFORMATTED',RECL=LENR)
C
      DO 10 ICH=1,NCH
         IA = 1
         IB = 2
         READ(IN,REC=ICH) (IDAT(I,IA),I=1,NPTB)
         DO 20 IBLK=2,NBLKS
            READ(IN,REC=ICH+(IBLK-1)*NCH) (IDAT(I,IB),I=1,NPTB)
            IDAT(NPTB,IA) = (IDAT(NPTB-1,IA) + IDAT(1,IB)) / 2
            WRITE(IOUT,REC=(IBLK-1)+(ICH-1)*NBLKS) (IDAT(I,IA),I=1,NPTB)
            CALL SWVAL( IA,IB )
20       CONTINUE
         IDAT(NPTB,IA) = IDAT(NPTB-1,IA)
         WRITE(IOUT,REC=NBLKS+(ICH-1)*NBLKS) (IDAT(I,IA),I=1,NPTB)
10    CONTINUE
C
      CLOSE(IN)
      CLOSE(IOUT)
C
      RETURN
      END
C
C
C
      SUBROUTINE CHKCMD( IRUN,IRUNCH,DRIVE)
      INTEGER*2 IRUN, NARGS, ARGC, LENR, CLEN, LENCH
      CHARACTER IRUNCH*3, DRIVE*2, ARGV*8, CH1*1, CH6*6, CH8*8
C
      NARGS = ARGC() - 1
      IF(NARGS.NE.3) THEN
         WRITE(*,*) 'The calling sequence is:'
         WRITE(*,*) '    GETHFF   run#   drive_letter   /R2048'
         STOP
      ENDIF
      CH8   = ARGV(1)
      LENCH = CLEN(CH8 )
      IF(LENCH.GT.3) STOP 'The second command-line arguement is limited
     * to 3 characters.'
C
      READ(CH8,'(I8)') IRUN
      IF(IRUN.LE.9) THEN
         WRITE(IRUNCH,'(2H00,I1)') IRUN
      ELSE IF(IRUN.LE.99) THEN
         WRITE(IRUNCH,'(1H0,I2)') IRUN
      ELSE IF(IRUN.LE.999) THEN
         WRITE(IRUNCH,'(I3)') IRUN
      ELSE
```

```
         STOP 'The run number is limited to 3 digits.'
      ENDIF
C
      WRITE(*,102) IRUN, IRUNCH
102   FORMAT('IRUN = ',I3,5X,'IRUNCH = ',A3)
C
      CH1 = ARGV(2 )
      IF((CH1.EQ.'A').OR.(CH1.EQ.'a')) THEN
         DRIVE = 'A:'
      ELSE IF((CH1.EQ.'B').OR.(CH1.EQ.'b')) THEN
         DRIVE = 'B:'
      ELSE IF((CH1.EQ.'C').OR.(CH1.EQ.'c')) THEN
         DRIVE = 'C:'
      ELSE
         STOP 'The second com-line argument must be the output disk
     * drive A, B, or C.'
      ENDIF
C
      CH6 = ARGV(3 )
      IF(CH6.NE.'/R2048') STOP 'The third com-line arg must be  /R2048'
C
      RETURN
      END
C
C
C
      SUBROUTINE SWVAL( IA,IB )
C
C     This subroutine interchanges the values of IA and IB.
C
      INTEGER*2 IA, IB, ISTORE
      ISTORE = IA
      IA     = IB
      IB     = ISTORE
      RETURN
      END
```

APPENDIX I

SOURCE CODE FOR THE GRAPHIC DATA DISPLAY MODULE

All of the C source code for the graphic data display was compiled and linked under Microsoft C version 5.1. The compiler and linker options as well as the list of source files are indicated below in the "make" files DSHI.MK and DSLO.MK. The procedure files, the "make" files, and all of the C source files are stored in directory APPNDX_I on the enclosed diskette.

FILE: DSH.BAT

```
MSHERC
DEL F:H0??.DAT
COPY A:H%1.DAT F:
DSHI F:H%1.DAT
```

FILE: DSL.BAT

```
MSHERC
DEL L0??.DAT
COPY A:L%1.DAT
DSLO L%1.DAT
```

FILE: DSHI.MK

```
dshi.obj: dshi.c
          cl /c dshi.c

readb.obj: readb.c
          cl /c readb.c

rect.obj: rect.c
          cl /c rect.c

tmshft.obj: tmshft.c
          cl /c tmshft.c

ypix.obj: ypix.c
          cl /c ypix.c

maxab.obj: maxab.c
          cl /c maxab.c

gulp.obj: gulp.c
          cl /c gulp.c

dshi.exe: dshi.obj readb.obj rect.obj tmshft.obj ypix.obj maxab.obj gulp.obj
cl /F 8000 dshi.obj readb.obj rect.obj tmshft.obj ypix.obj maxab.obj gulp.obj
```

FILE: DSLO.MK

```
dslo.obj: dslo.c
          cl /c dslo.c

readb.obj: readb.c
          cl /c readb.c

rect.obj: rect.c
          cl /c rect.c

tmshft.obj: tmshft.c
          cl /c tmshft.c

ypixn.obj: ypixn.c
          cl /c ypixn.c

minmax.obj: minmax.c
          cl /c minmax.c

dset.obj: dset.c
          cl /c dset.c

dslo.exe: dslo.obj readb.obj rect.obj tmshft.obj ypixn.obj minmax.obj dset.obj
cl /F 8000 dslo.obj readb.obj rect.obj tmshft.obj ypixn.obj minmax.obj dset.obj
```

APPENDIX J

SOURCE CODE FOR THE DATA ANALYSIS MODULE

The data analysis module consists of only one main program FLUX1, but because of its size its source code is contained in eight files: FLUX1.FOR, OUT1.FOR, OUT2.FOR, ... , OUT7.FOR. All of the source files were compiled under the Ryan-McFarland FORTRAN compiler version 2.43 using compiler options /LIZY.

Below are listed the procedure file FLUXA.BAT and a portion of file FLUX1.FOR . All of the source files in their entirety are given in directory APPNDX_J of the enclosed diskette.

FILE: FLUXA.BAT

```
DEL A:F0??.V1?
COPY A:P%1.DAT
COPY A:H%1.DAT
COPY A:L%1.DAT
FLUX1 %1 /R16384
COPY F%1.V19 A:
DEL H%1.DAT
DEL L%1.DAT
PAUSE
```

FILE: FLUX1.FOR

```
      PROGRAM FLUX1
C
C     Programmed by J T Priestley July 14-Aug xx,1988.
C
C     This program processes data from the fall 1988 heat-momentum
C     vertical flux experiment at Table Mountain.
C
      PARAMETER (NHPTB=8192, NLPTB=512, NBD=5, NLOCHD=12)
C
C        NHPTB = The Number of High freq PoinTs per Buffer for one
C                channel.
C        NLPTB = The Number of Low freq PoinTs per Buffer for one
C                channel.
C        NBD   = The dimension (5) of the arrays used to store averaged
C                quantities. The first four elements store the averages
C                of the four quarters of each time series and the last
C                element stores the overall average.
C        NLOCHD= The second dimension (12) of array IDAT and thus the
C                maximum number of low freq channels.
C
C
C     Sept. 11,1988: Modified to Version 1.1
```

```
C       The sign of the propvane angle VAN(I) in subroutine ENGL was
C       reversed so that the wind angle from the vane now increases as
C       the wind vector rotates counter clockwise (as viewed from above).
C       This makes the vane angle--relative to the boom--correspond to the
C       established convention of the sonic anemometer (Kaimal, J.C.,1980,
C       Sonic Anemometers, edited by F. Dobson, L,Hasse, and R.Davis,
C       Plenum Publishing Corp. N.Y.).  To accomodate this change, and to
C       correct the error in the previous sonic geographic wind angle the
C       geographic wind angle is now computed as the boom angle minus
C       the wind angle from the sonic or propvane.
C
C
C       Oct. 6,1988: Modification to Version 1.2
C       This modification makes the following significant changes:
C          1. Corrects the computation of TEMPSP the TEMPerature Scaling
C             Parameter in file OUT1.FOR subroutine OUTFLUX.
C          2. Makes a major addition to subroutine OUT4, and
C          3. Adds OUT5, OUT6, and OUT7.
C
C
C       Oct. 17,1988: Modification to Version 1.3
C       This modification makes some cosmetic changes to the screen
C       output, a format change to the OUT1 output, and an update to the
C       source-code comments.
C
C
C       Oct. 20,1988: Modification to Version 1.4
C       This modification changes a line of code in subroutine ENGL from
C           CV2(I)  = 388.1  *  (COEF(6)*IDAT(I,6))**2
C       to
C           CV2(I)  = 388.1  *  ((COEF(6)*IDAT(I,6)-.0015)**2 - .00002809)
C       This change is intended to correct for measured imperfections in
C       the rms-to-dc converter in the CV2 channel.
C       ****************************************************************
C       *   Note: This change and therefore this version (1.4) applies  *
C       *         only to run numbers 1 through 16.                     *
C       ****************************************************************
C
C
C       Oct. 21,1988: Modification to Version 1.5
C       Starting with run number 17 and for all subsequent runs we
C       inserted a precision 'times 10' amplifier before the rms-to-dc
C       converter to diminish the effect of the imperfect converter.
C       (This effect is much more pronounced at lower amplitudes.)
C       To accomodate this change the pertinent line of code in subroutine
C       ENGL was changed to:
C           CV2(I)  = 3.881  *  ((COEF(6)*IDAT(I,6)-.0015)**2 - .00002809)
C       ****************************************************************
C       *   Therefore Version 1.5 applies to run numbers 17 and after.  *
C       ****************************************************************
```

```
C
C
C      Jan. 5,1989: Modification to Version 1.6
C      This version combines the effects of versions 1.4 and 1.5 into one
C      version by testing for the run number, and then calling alternate
C      versions of subroutine ENGL.  Subroutine ENGL_A is taylored for
C      runs 1 through 16, while subroutine ENGL_B is for runs 17 through
C      52.  Also--and the thing that precipitated this new version of
C      FLUX1--, we discovered in looking back at the log book that the T′
C      sensor was hooked up with reversed polarity for runs 1 and 2.
C      To correct this error we are splitting subroutine ENGH into two
C      versions--ENGH_A for runs 1 and 2, and ENGH_B for runs 3 through
C      52.
C
C
C      Feb. 2,1989: Modification to Version 1.7
C      This version corrects two errors in previous versions.
C      The Monin-Obukhov length was not being properly computed in
C      subroutine OUT1 because the variable TCBAR was not being passed
C      into the subroutine.  The other error was in subroutine OUT6.
C      The quantity <w′**2>/frict_vel**2 was not being properly computed
C      because frict_vel was not being squared.
C
C
C      Version 1.8:  April 11,1989, April 20,1989.
C      This version corrects two errors in the program--one in subroutine
C      SONIC1 and one in OUT3--and adjusts the values of some constants
C      in subroutine OUT4 and function GFUN.  On April 20,1989 I modified
C      the section titles output by subroutines OUT1, OUT4, OUT5, and
C      OUT6.
C
C
C      Version 1.9:  August 7,1989
C      This version changes the Section 7. (FLUXES FROM PROPAGATION)
C      results so that they are based on the inner-scale scintillometer
C      (150-meter path) rather than the long-path (607-meter)
C      scintillometer.  This changes the calling sequences (in this
C      program) of subroutines OUT2, OUT3, and OUT7.  Subroutine OUT2
C      now outputs, and subroutine OUT3 inputs the variable CN2LA_RI_IS.
C      Also, the last formal parameter in subroutines OUT3 and OUT7
C      was changed from CT2_OPT_HI to CT2_OPT_LA.  Subroutine OUT3 was
C      changed to compute the new parameter, and in subroutine OUT7 a
C      simple variable name-change is made.
C
C
C      In subroutine SONIC1 (file FLUX2.FOR) the statement
C           IF(ATHETA.LT.R75) AI = AI*(DRAG + SLOPE*ATHETA)
C      was changed to
C           IF(ATHETA.LT.R75) AI = AI/(DRAG + SLOPE*ATHETA)
C      and its companion statement (following) was similarly changed.
```

```
C
C      In subroutine OUT3 (file OUT3.FOR) the statements
C             NDIM_CT2_LO(I)     = ZH**F23*CT2LO/TEMPSP(I)**2
C      and    NDIM_CT2_OPT_LO(I) = ZH**F23*CT2_OPT_LO/TEMPSP(I)**2
C      were corrected to
C             NDIM_CT2_LO(I)     = ZL**F23*CT2LO/TEMPSP(I)**2
C      and    NDIM_CT2_OPT_LO(I) = ZL**F23*CT2_OPT_LO/TEMPSP(I)**2 .
C
C      In subroutine OUT4 (file OUT4.FOR) the constant '2.2' in the
C      statement  ZDEN = ....  was changed to '2.75'.
C      Also the constant '6.1' in the statement ZETA_X(I) = ...
C      was changed to '7.0 .
C
C      In function GFUN (file OUT7.FOR) the DATA statements
C             DATA A,B/ 4.9, 8.1, 17., 6.1, 13., 46. /
C             DATA C,D/ 4.9, 2.5, 2.5, 2.2, 4.3, 4.3 /
C      were changed to
C             DATA A,B/ 4.9, 8.1, 17., 7.0, 15., 46. /
C             DATA C,D/ 4.9, 2.7, 2.7, 2.75, 5.0, 5.0 / .
C
C
C      High freq arrays and parameters.
       INTEGER*2 IA(NHPTB), IB(NHPTB), IW(NHPTB), IT(NHPTB), LENRH,
     *           IDATBEG(3), ITIMBEG(4), NHICH
       INTEGER*4 NHPTT
       REAL A(NHPTB), B(NHPTB), W(NHPTB), T(NHPTB), U(NHPTB), V(NHPTB)
       REAL UAV(NBD), VAV(NBD), WAV(NBD), TAV(NBD), UW(NBD), VW(NBD),
     *     WW(NBD), TW(NBD), TT(NBD), SAV(NBD)
       REAL PAG(16), HISTA(-7:7,NBD), DATHI(NBD,10)
       CHARACTER IRUNCH*3, HNAME*8
       EQUIVALENCE (UAV(1),DATHI(1,1)), (VAV(1),DATHI(1,2)),
     *             (WAV(1),DATHI(1,3)), (TAV(1),DATHI(1,4)),
     *             (UW(1), DATHI(1,5)), (VW(1), DATHI(1,6)),
     *             (WW(1), DATHI(1,7)), (TW(1), DATHI(1,8)),
     *             (TT(1), DATHI(1,9)), (SAV(1),DATHI(1,10))
C
C
C      Low freq arrays and parameters.
       INTEGER*2 IDAT(NLPTB,NLOCHD), LENRL, NLOCH, NLPTT
       REAL CN2LO(NLPTB), CN2HI(NLPTB), CN2LSR(NLPTB), CN2LA(NLPTB),
     *     WOPT(NLPTB), CV2(NLPTB), CT2LO(NLPTB), CT2HI(NLPTB),
     *     PROP(NLPTB), VAN(NLPTB), BARM(NLPTB), ANGL(NLPTB),
     *     UPV(NLPTB), VPV(NLPTB)
       REAL HISTVAN(-7:7,NBD), DATLOW(NBD,13), FRICTVEL(3), TEMPSP(3),
     *     MON$OBK(3), ND_CT2(4,3)
       CHARACTER LNAME*8, FNAME*8
C
C      VERSION NUMBER
C      VER = 1.0
C      VER = 1.1
```

```
C      VER = 1.2
C      VER = 1.3
C      VER = 1.4
C      VER = 1.5
C      VER = 1.6
C      VER = 1.7
C      VER = 1.8
       VER = 1.9
C
C      Get high and low freq parameters.
       LENRH = 2*NHPTB
       LENRL = 2*NLPTB
       DATA WAVLNLSR, PATHLSR / .6328E-6, 150. /
       CALL CHKCMD(LENRH,LENRL, IRUN,IRUNCH)
       CALL INPAR(IRUNCH, TCBAR,ZH,ZL,RHBAR,IDATBEG,ITIMBEG,
     *            NHICH,NLOCH,NHPTT,NLPTT,PAG)
CD     WRITE(*,*) 'FLUX1: ',TCBAR,ZH,ZL,RHBAR,IDATBEG,ITIMBEG,
CD   *                    NHICH,NLOCH,NHPTT,NLPTT,PAG
       CALL CHKPTS(NHPTT,NHPTB,NLPTT,NLPTB,NBD, NBLKS)
C
C      Now process high freq data.
C
       WRITE (HNAME,'(1HH,A3,4H.DAT)') IRUNCH
       IU = 10
       OPEN(UNIT=IU,STATUS='OLD',FILE=HNAME,ACCESS='DIRECT',
     *      FORM='UNFORMATTED',RECL=LENRH)
       WRITE(*,*) 'Begin processing high frequency data.'
       DO 10 I=1,NBLKS
          WRITE(*,*) '    Begin block # ',I
          CALL INHDAT(IU,I,NHICH,NHPTB,NBLKS, IA,IB,IW,IT)
          IF(IRUN.LE.2) CALL ENGH_A(IA,IB,IW,IT,TCBAR,NHPTB,PAG, A,B,W,T)
          IF(IRUN.GE.3) CALL ENGH_B(IA,IB,IW,IT,TCBAR,NHPTB,PAG, A,B,W,T)
          CALL SONIC1(A,B,NHPTB, U,V,HISTA(-7,I))
          CALL COMPH(U,V,W,T,NHPTB, UAV(I),VAV(I),WAV(I),TAV(I),
     *               UW(I),VW(I),WW(I),TW(I),TT(I),SAV(I))
10     CONTINUE
       CLOSE(IU)
C      Compute final high freq averages and place in last array positions
       CALL COMPH1(NBLKS,NBD, UAV,VAV,WAV,TAV,UW,VW,WW,TW,TT,SAV,HISTA )
C
C      Now process low freq data.
C
       WRITE(LNAME,'(1HL,A3,4H.DAT)') IRUNCH
       IU = 11
       OPEN(UNIT=IU,STATUS='OLD',FILE=LNAME,ACCESS='DIRECT',
     *      FORM='UNFORMATTED', RECL=LENRL)
       WRITE(*,*)
       WRITE(*,*) 'Begin processing low frequency data.'
       DO 20 I=1,NBLKS
          WRITE(*,*) '    Begin block # ',I
```

```
        CALL INLDAT(IU,I,NLOCH,NLPTB,NBLKS, IDAT)
        IF(IRUN.LE.16) CALL ENGL_A(IDAT,NLPTB,NHICH,NLOCH,PAG, CN2LO,
     *                             CN2HI,CN2LSR,CN2LA,WOPT,CV2,CT2LO,
     *                             CT2HI,PROP,VAN,BARM,ANGL)
        IF(IRUN.GE.17) CALL ENGL_B(IDAT,NLPTB,NHICH,NLOCH,PAG, CN2LO,
     *                             CN2HI,CN2LSR,CN2LA,WOPT,CV2,CT2LO,
     *                             CT2HI,PROP,VAN,BARM,ANGL)
        CALL PRPVAN(PROP,VAN,NLPTB, UPV,VPV,HISTVAN(-7,I))
        CALL COMPL(CN2LO,CN2HI,CN2LSR,CN2LA,WOPT,CV2,CT2LO,CT2HI,
     *             PROP,UPV,VPV,BARM,ANGL,NLPTB,NBD,I, DATLOW )
20    CONTINUE
      CLOSE(IU)
C     Compute final low freq averages and place in last array positions.
      CALL COMPL1(NBLKS,NBD, DATLOW,HISTVAN )
C
C     Now analyze and output data into file Fxxx.Vyy, where xxx is the
C     run number and yy is the version number times 10.
C
      WRITE(FNAME,'(1HF,A3,2H.V,I2)') IRUNCH, IFIX(10.*VER+.01)
      IU = 12
      OPEN(UNIT=IU,STATUS='NEW',FILE=FNAME,ACCESS='SEQUENTIAL',
     *     FORM='FORMATTED')
C
      WRITE(*,*)
      WRITE(*,*) 'Now begin final computations and output.'
      ANGLE = DATLOW(NBD,13)
      PBAR  = DATLOW(NBD,12)
      WRITE(*,*) '*** ENTERING OUTHDR ***'
      CALL OUTHDR(IU,VER,IRUNCH,TCBAR,PBAR,RHBAR,ZH,ZL,ANGLE,IDATBEG,
     *            ITIMBEG )
      CALL LINES(IU,4 )
C
      WRITE(*,*) '*** ENTERING OUT1 ***'
      CALL OUT1(IU,NBD,TCBAR,DATHI,HISTA,DATLOW,HISTVAN, FRICTVEL,
     *          TEMPSP,MON$OBK)
      CALL LINES(IU,7 )
C
      WRITE(*,*) '*** ENTERING OUT2 ***'
      CALL OUT2(IU,DATLOW,NBD,ZH,ZL,WAVLNLSR,PATHLSR,FRICTVEL,MON$OBK,
     *          TCBAR,PBAR, EPS_OPT_RI,CN2LA_RI_IS)
      CALL LINES(IU,4 )
C
      WRITE(*,*) '*** ENTERING OUT3 ***'
      CALL OUT3(IU,DATLOW,NBD,ZH,ZL,TEMPSP,MON$OBK,TCBAR,PBAR,
     *          CN2LA_RI_IS, ND_CT2,CT2_OPT_LA)
      CALL LINES(IU,13 )
C
      WRITE(*,*) '*** ENTERING OUT4 ***'
      CALL OUT4(IU,DATLOW,NBD,ZH,ZL,MON$OBK,ND_CT2 )
      CALL LINES(IU,4 )
```

```
C
      WRITE(*,*) '*** ENTERING OUT5 ***'
      CALL OUT5(IU,DATLOW,NBD,ZH,FRICTVEL,MON$OBK )
      CALL LINES(IU,4 )
C
      WRITE(*,*) '*** ENTERING OUT6 ***'
      CALL OUT6(IU,NBD,ZH,DATHI,FRICTVEL,TEMPSP,MON$OBK )
      CALL LINES(IU,3 )
C
      WRITE(*,*) '*** ENTERING OUT7 ***'
      CALL OUT7(IU,ZH,TCBAR,EPS_OPT_RI,CT2_OPT_LA )
C
      STOP
      END
C
C
C
      SUBROUTINE CHKCMD(LENRH,LENRL, IRUN,IRUNCH)
C
C     This subroutine checks for the proper command-line calling
C     sequence for the main program, and returns the run number in
C     both INTEGER*2 and CHARACTER*3 formats.
C
      INTEGER*2 NARGS, ARGC, LENRH, LENRL, LENCH, CLEN, I1, I2
      CHARACTER CH7*7, CH8*8, IRUNCH*3, LENRHCH*4, ARGV*8
      I1 = 1
      I2 = 2
      NARGS = ARGC() - 1
      IF(NARGS.LT.2) CALL MESSG(1,LENRH )
      CH8 = ARGV(I1)
      LENCH = CLEN(CH8 )
      IF(LENCH.GT.3) CALL MESSG(2,LENRH )
      READ(CH8,'(I8)') IRUN
C
      IF(IRUN.LE.9) THEN
         WRITE(IRUNCH,'(2H00,I1)') IRUN
      ELSE IF (IRUN.LE.99) THEN
         WRITE(IRUNCH,'(1H0,I2)') IRUN
      ELSE IF (IRUN.LE.999) THEN
         WRITE(IRUNCH,'(I3)') IRUN
      ELSE
         STOP 'sub CHKCMD: The run number IRUN is limited to 3 digits.'
      ENDIF
C
      WRITE(*,101) IRUN, IRUNCH
101   FORMAT('IRUN = ',I3,5X,'IRUNCH = ',A3)
      CH7 = ARGV(I2)
      IF(CH7.NE.'/R16384') STOP 'sub CHKCMD: The second command-line par
     *ameter must be ''/R16384'' with no spaces.'
      IF(LENRH.NE.16384) STOP 'sub CHKCMD: LENRH must equal 16384'
      IF(LENRL.NE.1024)  STOP 'sub CHKCMD: LENRL must equal 1024'
C
      RETURN
      END

          .
          .
          .
```

(The remainder of this file along with files OUT1.FOR, OUT2.FOR, ... OUT7.FOR are on the diskette in directory APPNDX_J.