General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

Produced by the NASA Center for Aerospace Information (CASI)
A PORTABLE REAL-TIME DATA PROCESSING SYSTEM FOR STANDARD METEOROLOGICAL RADIOSONDES

Final Technical Report

under

Research Grant NAG6-15
Meteorological Data Processing Firmware

for the period

May 15, 1981 to November 30, 1982

Forrest L. Stafranso
Principal Investigator

University of Utah
College of Engineering
Department of Electrical Engineering
Salt Lake City, Utah 84112
FOREWORD

The portable and complete meteorological data processing system described herein provides user-ready hardcopy in the field, essentially simultaneously at the time and place of the ascent of a conventional radiosonde balloon. Its development has been an exciting application of both the automatic meteorological data processing concepts generated over the years at the University of Utah, and of the portable computing power now available through recent microprocessor technology.

Acknowledgements are due several undergraduate electrical engineering students, who produced UMET-1. Much of the preliminary work which established feasibility was done by Daniel G. Schmidt before his graduation in June 1981. Anthony C. Barkans during his senior year designed the hardware and made the software innovations necessary to adapt the available Motorola 6809 FORTRAN capability to this real-time application. His simple masking technique for automatic synchronizing to the biphase serial input signal, his hardware approach to synchronizing to the 100-bit input record from the TRADAT receiver system in order to conserve computing time, and his ROM board modification to provide an economical two-page memory map are among the measures taken to produce a working system. A student associate, Bijan Yadegar, provided valuable assistance, particularly in programming MONITOR, and in implementing the ROM bank-switching modification. Tony Henderson designed the punched paper tape reader interface as part of his senior thesis project. The project team is grateful for the "extra mile" afforded by Clint Bauer of Motorola Semiconductor Products at times of special need for equipment and information. Finally, Chris L. Burks contributed considerable time
and effort in documenting the project, following the intensive and long hours of debugging and hardware changes conducted by his predecessors in the final weeks before the first successful complete test runs of the system.

Gratitude is most sincerely expressed to W. W. West, R. V. Snyder, and their associates at NASA Wallops Flight Center for their support and cooperation, and to the staff of the Electrical Engineering Department at the University of Utah for their excellent service in support of the project.

As with most "originals", tempting improvements and further advances are obvious; nevertheless, it is felt that UMET-1 represents a significant step forward in meteorological operations capability.
# TABLE OF CONTENTS

1.0 INTRODUCTION .................................................. 1
  1.1 Printer ...................................................... 4
  1.2 Keyboard - Video Terminal .................................. 5
  1.3 Memory Map .................................................. 6

2.0 PROCESSOR HARDWARE ............................................ 9
  2.1 Motherboard .................................................. 9
  2.2 Paper Tape Reader Head .................................... 12
  2.3 Input Board .................................................. 14
    2.3.1 Biphase-to-Level Converter ......................... 14
    2.3.2 Paper Tape Interface ................................ 16
  2.4 Interface Board ............................................. 18
    2.4.1 The Microprocessor Side of the Interface ........ 18
    2.4.2 The Real-Time Side of the Interface .............. 19
    2.4.3 Punched Paper Tape Side of the Interface ...... 25
  2.5 Read Only Memory (ROM) Board ............................ 25
  2.6 Random Access Memory (RAM) Board ....................... 27
  2.7 Central Processor Unit (CPU) Board ..................... 27

3.0 PROCESSOR SOFTWARE ............................................ 29
  3.1 Interrupt Design ............................................ 31
    3.1.1 The Queue, Real-Time Buffer ..................... 34
    3.1.2 Interrupts While Advancing the Queue ........... 34
    3.1.3 Manual Stop, STOPER ............................... 35
  3.2 MONITOR ..................................................... 35
    3.2.1 Interrupt Error Message ......................... 38
  3.3 AMET ......................................................... 38
    3.3.1 MAIN ................................................ 40
    3.3.2 PRINTER .............................................. 41
    3.3.3 SETUP .............................................. 42
    3.3.4 TAPE ................................................ 43
    3.3.5 STOPER ............................................. 45
    3.3.6 OFF ................................................ 45
    3.3.7 INTERR .............................................. 45
    3.3.8 ADVANC ............................................ 47
    3.3.9 REAL ............................................... 48
    3.3.10 GETBK .............................................. 49
    3.3.11 ANGLE ................................................ 49
    3.3.12 TRACK ............................................. 50
    3.3.13 SEARCH .............................................. 50
    3.3.14 DECOM ............................................... 51
    3.3.15 INTERP ............................................. 51
    3.3.16 JUMPER .............................................. 51
  3.4 BMET ......................................................... 52
    3.4.1 PRDMAIN ............................................. 52
    3.4.2 PRD ................................................. 52
    3.4.3 TEMPCE ............................................... 53
    3.4.4 RL ................................................ 53
    3.4.5 WINDS .............................................. 54
    3.4.6 Added FORTRAN functions ......................... 54
4.0 OPERATION OF UMET-1 .............................................. 55
  4.1 Equipment Set Up .............................................. 55
  4.2 Operating Procedure .......................................... 57
  4.3 Description of Video Display Output ......................... 60

5.0 CONCLUSIONS AND RECOMMENDATIONS ......................... 63

REFERENCES .......................................................... 65

APPENDICES
  A. UMET-1 Program Listing
  B. Sample Output Listing
  C. Circuit Diagrams
  D. Future Application to Rocketsonde Data: METROC-R
1.0 INTRODUCTION

UMET-1 is a microprocessor-based portable system for automatic real-time processing of flight data transmitted from the standard RAWINSONDE upper atmosphere meteorological balloonsonde. This report describes the first "target system," delivered for initial operational use. This first system is designed to receive data from a mobile tracking and telemetry receiving system (TRADAT), as the balloonsonde ascends to apogee. UMET-1 automatically processes the data in real-time, and produces, after balloon-burst, user-ready hardcopy. The listing includes, at one-minute intervals of the ascent and at selected pressure levels, the following measured and derived meteorological quantities:

- Altitude (geopotential meters)
- Pressure (mb)
- Logarithm of pressure
- Temperature (K)
- Virtual temperature (K)
- Relative humidity (percent)
- Dewpoint (K)
- Specific humidity
- Wind speed (m/sec)
- Wind direction (degrees azimuth)
- Wind, northerly (m/sec)
- Wind, easterly (m/sec)

UMET-1 is unique among automatic meteorological data processing systems, in that it does not require a specially designed sonde. The processing includes decommutation of the standard sonde signal into reference, temperature, and humidity channels.
The system is composed of the three portable units shown in Figure 1.0-1: the central unit (card cage containing five printed circuit boards, and with a punched paper tape reader head mounted on its front panel), a keyboard-video terminal, and a printer. Upon setting up the three units with two interconnecting cables (C1, C2), and connecting to the input data source and to a standard 120 VAC power source, the system is ready for operation.

The operator simply turns on the system, and enters keyboard inputs as requested on the video screen. A "pull-through" punched paper tape reader accepts the pressure calibration data routinely supplied by the sonde manufacturer. The system automatically commences processing when the time-word on the incoming data exceeds the launch time entered by keyboard. Provision is included for correction of keyboard errors, and for changing the entered launched time, in case of delays in balloon release. Certain computed quantities are displayed during balloon ascent to confirm normal operation of real-time processing. Upon completion of printing the output, the system automatically "idles" (enters an infinite loop) and waits to be turned off or reinitialized for inputs for the next balloon launch.

The computed output is that of the NASA Wallops Computer Program No. 3.0.0700 ECC-PRD [1], excluding the ozone portions. The data-editing, -condensing, -decommutating, and -interpolating functions, as well as the baroswitch tracking function, are those of the University of Utah program RAWINPROC [2]. Appendix A is a complete listing of the system software, including the modified versions of the above FORTRAN programs and the assembly-language programs, used in UMET-1.
Fig. 1.0-1. UMET consists of the three portable units: the central unit (containing the printed-circuit boards, a paper tape reader, and power supply), a keyboard-video terminal, and a printer. Real-time data are received from the TRADAT system at the "DATA-IN" jack on the front panel. Baro-switch calibration data are read by the punched paper tape reader head. Flat cables C1 and C2 (not shown) connect the keyboard-video terminal and printer, respectively, to the central unit. Each of the three units require 120 VAC power.
The remainder of this introductory chapter will identify the commercially available input and output units (printer and keyboard terminal), and will give the memory map of the processor, as an aid to reading the next two chapters. Chapter 2.0 will describe the hardware of the processor and Chapter 3.0 the software. Chapter 4.0 presents a step-by-step procedure for setting up and operating UMET-1 and describes the auxiliary quantities displayed on the video terminal during and after real-time processing. Finally, Chapter 5.0 suggests improvements and further applications.

Appendix A, the UMET-1 computer routines, is followed by Appendix B, a sample output listing. Appendix C contains diagrams and drawings of the hardware system.

The software for a future application of the UMET concept, that of meteorological rocket (DATASONDE) flight data, is listed in Appendix D. Though this program requires slant range data, which, in turn, require a transponder in the parachutesonde (unavailable at present), the UMET concept accommodates this application. Further work is required to complete the incorporation and demonstration of a real-time version of the METROC data reduction routine.

1.1 Printer

The principal output unit of UMET-1 is a standard 100 character per second dot-matrix printer (CENTRONICS Model 739-1) [3]. Its pin-addressable graphics capability, through host computer control, accommodates the recommended addition to the system of a plotter routine, which would produce the familiar "pen trace" produced by the conventional
AN/TMQ-5 meteorological recorder. This pen trace would serve not only as a familiar real-time monitoring feature, but would serve as a backup under unusual conditions. The printer provides a permanent record of each run including inputs entered manually (by keyboard) or by the punched paper tape reader, and optional diagnostic quantities during the processing, as well as the ultimate output, the printed results of the sounding.

The 40-pin printed circuit edge card connector on the left rear of the printer, connects a 36-wire flat cable C2 through a special adapter (Motorola printer cable board) to the UMET-1 central unit. The adapter receives the printer cable in a 36-pin D connector (Amphenol 57-10360-13). From a 3M No. 3426 connector, a 50-wire flat cable then leads inside the central unit to edge connector P4 on the CPU Board.

1.2 Keyboard-Video Terminal

The keyboard terminal is a Televideo Model 910 [4]. The keyboard is selectric style and includes a ten-key pad for easy entry of numbers. The CRT display screen is used to request successive input values and decisions from the operator, and to display incidental diagnostic and status information during processing. The original UMET design excluded the CRT display for economy and ruggedness, and used only the keyboard and printer for input and output. However, the computer terminal chosen was sufficiently inexpensive and rugged, and at the same time offered added convenience, for the prototype system. Subsequent models for field use may well exclude the video screen. The terminal is used in the system without modification and is therefore interchangeable.
Switch settings on the terminal, for use in UMET-1 are listed in Table 1.2-1. A 25-wire flat cable Cl from connector P3 (RS-232) on the terminal, leads to edge connector P2 on the CPU Board.

1.3 Memory Map

The remaining one of the three units mentioned above comprising UMET-1, the central unit, is described in the following two chapters, in terms of hardware and software. The system hardware is configured so that the software sees the system as shown in Figure 1.3-1. Throughout this report the prefix $ is used to denote hexadecimal, i.e. base 16, numbers.

The ROM banks are software selectable so that bank A is used during real-time processing and bank B is used to complete the meteorological computations and to list output immediately following the termination of the processing of flight input data. Of the 64K ($FFFF, or 65,536 addresses) available memory space, one-half or 32K ($8000 or 32,768 addresses) is reserved for ROM, 24K ($6000, or 24,576 addresses) for RAM, and the remaining one-eighth or 8K ($2000, or 8,192 addresses) is designated for input-output interfaces and the program monitor. The largest part, that for ROM, holds either of two programs "AMET" or "BMET", depending on which bank is switched in by the monitor. It is noted that the additional 2K of RAM on the CPU board is unused (see Section 2.7 below).
**TABLE 1.2-1. TERMINAL SWITCH SETTINGS FOR UMET-1**

<table>
<thead>
<tr>
<th>Switch S1</th>
<th>Switch S2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 D</td>
<td>1 D</td>
</tr>
<tr>
<td>2 D</td>
<td>2 D</td>
</tr>
<tr>
<td>3 D</td>
<td>3 U</td>
</tr>
<tr>
<td>4 D</td>
<td>4 D</td>
</tr>
<tr>
<td>5 D</td>
<td>5 D</td>
</tr>
<tr>
<td>6 D</td>
<td>6 U</td>
</tr>
<tr>
<td>7 D</td>
<td>7 U</td>
</tr>
<tr>
<td>8 D</td>
<td>8 D</td>
</tr>
<tr>
<td>9 U</td>
<td>9 U</td>
</tr>
<tr>
<td>10 D</td>
<td>10 D</td>
</tr>
</tbody>
</table>

(U = up, D = down)
### Interrupt Vectors

<table>
<thead>
<tr>
<th>Interrupt Vectors</th>
<th>$FFFF</th>
<th>$FFF2</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Monitor and Interrupt Error Message</th>
<th>$FA37</th>
<th>$F800</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Terminal ACIA</th>
<th>$EC15</th>
<th>$EC14</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Printer Interface</th>
<th>$EC13</th>
<th>$EC10</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Terminal ACIA</th>
<th>$EC15</th>
<th>$EC14</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Paper Tape Reader PIA</th>
<th>$EO17</th>
<th>$EO14</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Real-Time Data PIAs</th>
<th>$EO13</th>
<th>$EO00</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>RAM</th>
<th>$DFFF</th>
<th>$8000</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ROM bank A (AMET)</th>
<th>ROM bank B (BMET)</th>
<th>$7FFF</th>
<th>$0000</th>
</tr>
</thead>
</table>

**Fig. 1.3-1.** Memory Map of UMET-1. At balloon apogee the ROM is switched from bank A to bank B. Memory addresses are in hexadecimal notation. (ACIA: Asynchronous Communication Interface Adapter; PIA: Parallel Interface Adapter; RAM: Random Access Memory; ROM: Read-Only Memory).
2.0 PROCESSOR HARDWARE

The central unit of the three-piece UMET-1 system is a chassis with five circuit boards, and with a punched paper tape reader. The chassis is a Motorola M68MMLCi Micromodule Long Chassis, containing a power supply, a ten-card rack, and motherboard [5]. The custom features of the motherboard and the added features on the front panel including the punched paper tape reader head, are described below, along with descriptions of the five circuit boards in the card cage.

The overall hardware configuration of the five boards and motherboard is shown in Figure 2.0-1. Note that two clock lines from the bi-phase-to-level converter (RT CLOCK) are used to drive the thirteen shift-registers since TTL fan-out is about ten. Two additional lines (BIT6, BIT5) utilize an available inverter on the INPUT board to obtain the inverse of bit 6 in the real-time data word. This is used in the interrupt decoder to synchronize on the input data word. These, together with the data line for the real-time input (RT DATA), the data clock, and interrupt lines for the paper tape input (PT DATA, PT CLOCK, PT INTREQ), and the ROM bank control line (BMET), constitute the nine custom connections for UMET-1 on the card-cage motherboard.

2.1 Motherboard

The card-cage is shown in Figure 2.1-1. The five printed-circuit boards must be placed in the cage in the order given in Table 2.1-1, beginning with position "1" nearest the front panel. Position 9 is unavailable because the wire-wrap pins of the INPUT Board extend into the space of position 9. Thus the ROM, INPUT, and INTERFACE boards,
Fig. 2.0-1. The hardware configuration of UMET-1 processor. Eight data lines are used from the Input to the Interface Board, and one from the Interface Board to the ROM Board. Addresses $0000$ to $7FFF$, and $8000$ to $DFFF$ are contained in the ROM and RAM Boards, respectively.
Fig. 2.1-1. The card-cage and motherboard. Position "1" of the card-cage is nearest the chassis front panel (right). Notice the custom connections hardwired on the motherboard between the headers at positions 7, 8, and 10. Cable C3, which connects the front panel to the Input Board (at edge connector P2), is shown lying behind the front panel. The power supply is beside the card-cage, and the cooling fan is seen on the rear panel (left).

TABLE 2.1-1. CIRCUIT BOARD POSITIONS

<table>
<thead>
<tr>
<th>Position</th>
<th>Printed Circuit Card or Board</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-6</td>
<td>CPU, RAM BOARDS</td>
</tr>
<tr>
<td>7</td>
<td>ROM BOARD</td>
</tr>
<tr>
<td>8</td>
<td>INTERFACE BOARD</td>
</tr>
<tr>
<td>9</td>
<td>(unavailable position)</td>
</tr>
<tr>
<td>10</td>
<td>INPUT BOARD</td>
</tr>
</tbody>
</table>
those having the custom data lines mentioned above, must occupy, as indicated, position 7-10. The CPU and RAM boards may be placed in any of the other positions.

The card cage motherboard provides the address, data, and control bus interconnections between the circuit boards, as well as the dc voltages from the chassis power supply. The motherboard has for custom use a 16-pin header for each card slot, connected to unused pins C, D, E, F, 25, 26, 27, and 28 on each 86-pin card slot connector (the other row of eight pins in each header is grounded).

Since nine data lines are needed to the interface board (eight are accommodated by the wire-wrap header), an unused ground line (pin B) was utilized between the interface and input boards. The ground line was cut to isolate it from its remote connections. The nine interboard connections are listed in Table 2.1-2.

2.2 Punched Paper Tape Reader Head

The punched paper tape reader head used on UMET-1 was adapted from an RP-9362 Tape Reader/Punch system manufactured by EECO, Electronics Engineering Company of California [6]. The electric motor drive was removed to make a "pull-through" reader mounted on the front panel of the UMET-1 central unit. The resulting fixture conveniently and reliably reads the one-inch wide eight-level tape [7], containing the baro-switch pressure calibration data, accompanying each RAWINSONDE. Figure C-1 in Appendix C shows the circuit diagram.

The reader head is photoelectric, employing nine light-emitting diodes (LEDs) to transmit light through the holes in the punched paper
### TABLE 2.1-2. CUSTOM DATA LINES ON THE CARD-CAGE MOTHERBOARD

<table>
<thead>
<tr>
<th>Input Board Pin</th>
<th>Interface Board Pin</th>
<th>ROM Board Pin</th>
<th>Signal Names (Fig. 2.0-1)</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>25</td>
<td>RT DATA</td>
<td>Real-time data</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>26</td>
<td>RT CLOCK</td>
<td>Real-time clock</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>27</td>
<td>RT CLOCK</td>
<td>Real-time clock</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>D</td>
<td>BIT6</td>
<td>Real-time bit 6</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>BIT6</td>
<td>Real-time bit 6, inverted</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>28</td>
<td>PT DATA</td>
<td>Paper tape data</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>PT CLOCK</td>
<td>Paper tape clock</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>PT INTREQ</td>
<td>Paper tape interrupt request</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>E</td>
<td>BMET</td>
<td>ROM bank select</td>
<td></td>
</tr>
</tbody>
</table>
tape, and nine phototransistors to sense the light. The current through
the LED array is adjusted to bias the phototransistors into full conduc-
tion for holes in the tape, and into cutoff for no holes. The outputs
of the phototransistors, eight parallel data signals, and one sprocket
hole signal, are brought to the Input Board through cable C3.

Although the reader head has a tape guide adjustment for different
tape sizes, the circuitry of the Input Board is designed only for the
standard eight-bit tape. The tape can be pulled through the reader head
virtually at any speed but, of course, without backing up. Procedure
for operating the punched tape reader head is included in Section 4.1.

2.3 Input Board

The Input Board is constructed on a Motorola MEX68WW Wire Wrap
Module, and contains custom-built circuitry to interface the real-time
data source and the paper tape reader to the microprocessor PIAs. It
consists of two separate and independent circuits; the biphase-to-level
converter and the paper tape interface. Figure C-4 in Appendix C shows
the component layout. The two circuits on the board are described
separately below.

2.3.1 Biphase-to-level Converter

Figure C-2 in Appendix C shows the circuit of the biphase-to-level
converter. The purpose of this circuit is to take the real-time data
signal from the front panel DATA-IN jack, convert it from its biphase
format to an ordinary serial bit stream (with "high" and "low" levels
being "1" and "0" respectively), and to feed it through the bus to the
shift register on the Interface Board. In addition, this circuit gener-
ates the clock for the shift register.
The real-time input signal from the TRADAT system is a 1000 bit per second biphase PCM serial stream. Binary "ones" and "zeroes" are therefore represented respectively by transitions high-to-low and low-to-high, clocked at one millisecond intervals. Note that infinite sequences of "ones" or of "zeroes" are indistinguishable unless the receiver is synchronized with the sender. Note also, however, that at any change from "zero" to "one", or "one" to "zero", there is no transition. This missing transition is used for synchronizing the biphase-to-level converter. Whenever an expected transition does not occur, the circuit automatically "moves up" a half-cycle, thus locking on to a proper transition.

The circuit accomplishes this automatic synchronization simply by masking any transition occurring within about 0.8 millisecond after the preceding transition. The next transitions the circuit sees after a missing transition, then, are always proper transitions.

The biphase input from the DATA-IN jack on the front panel through cable C3 is fed to a comparator U1, whose threshold is adjustable by potentiometer R4. This adjustment provides versatility in reading data.
from analog tape players. The comparator output is fed through a pair of inverters on U10, to an edge detector using U5 and an OR gate, U7-1. Thus when a transition, high-to-low or low-to-high, occurs in the input signal, a pulse occurs at pin 3 of the following AND gate, U8-2. Potentiometer R3 is used in the external timing control circuit of the 74LS12? one-shot U2-2 so that its output pulse remains high for a nominal 0.8 milliseconds. This output fed back through an inverter U10-3 prevents recognition of a transition for 0.8 milliseconds. Recognized transitions, however, each cause a clock pulse from U3-2 to gate the input signal level through U6-1. Since the U2-2 output is connected to the inverting input, pin 9, of U3-2, the resulting clock pulse from U3-2 occurs at the end of the 0.8 millisecond period, causing the signal gated through U6-1 to be the starting voltage level of the succeeding bit transition. A "fine" adjustment of R3, then, permits "tuning", i.e. the selection of the best point between transitions to gate the signal.

The clock pulse from U3-2, in addition to clocking the gate U6-1, also clocks the shift registers receiving the gated signal. The shift register clock pulse is delayed 0.05 milliseconds behind the gate clock by triggering at U3-1 on the pulse trailing edge (connecting to the inverting input, pin 1, of U3-1). This allows settling of the input before it is clocked into the shift register. Inverters U9 provide two clock lines to the shift registers for sufficient fanout.

2.3.2 Paper Tape Interface

Figure C-3 in Appendix C shows the circuit of the paper tape interface. The purpose of this circuit is to interface the paper tape reader
head to the microprocessor PIAs. The outputs of the phototransistors in the tape reader head (Section 2.2), eight parallel data signals (CH1 - CH8) and one sprocket hole signal (SPROCKET), are brought to the Input Board through cable C3. The paper tape interface senses from the sprocket hole signal when new data are available from the tape head. The data are then latched into a shift register and sent serially through a bus line to the PIA on the Interface Board. After transmission, an interrupt is sent to the microprocessor via the PIA, and the circuit is ready to accept more data.

All nine input lines from the paper tape head are first buffered through LM339 comparators, which have a reference voltage of about 1.5 volts. The reference voltage is adjustable through the bias potentiometers on the comparators' noninverting inputs. The outputs of the comparators are inverted, i.e. an output is a logic "zero" for a hole in the tape and logic "one" for no hole.

After the comparator, the sprocket signal is delayed 1 to 2 milliseconds by a resistor, capacitor, and diode network (R63, C20, D2), to insure that the data holes are centered over the phototransistors and the comparators have settled. The signal is then differentiated (R61, R62, C15, D1) and applied to a Schmitt trigger (U17). Note that diode D1 is in the differentiator to prevent the Schmitt trigger input pulse from falling below ground, which can cause spurious switching. The result is a signal called LATCH (Figure C-3), which goes high when the data are ready at the comparators and stays high for about a millisecond. On the rising edge of LATCH the data from the comparators are latched into the latches (U6), and on the falling edge they are transferred to the shift register.

- 17 -
At this point the circuit begins transmitting the data to the PIAs. The circuit includes a clock (U15) which runs at approximately 16 KHz and is used for shifting the data out of the shift register and into another shift register on the Interface Board. After the LATCH signal falls, a counter (U19) allows exactly eight clock pulses to reach the local shift register. The serial data signal, labelled PT DATA (Figure C-3), is sent through custom bus line 28. At the same time the clock signal is sent to the other shift register through custom bus line F, labelled PT CLOCK. At the conclusion of the eight pulses an interrupt is sent to the PIA through custom bus line B, labelled PT INTREQ. This interrupt tells the microprocessor that there are paper tape data to be read in from the PIA, which is connected to the other shift register. The current byte must be read from the PIA before the next byte arrives, or it is lost.

2.4 Interface Board

The Interface Board is built on a Motorola MEX68USM Universal Support Module [8]. This module has a wire-wrap socket area, address and control bus buffers, and switch selectable address capability, to accommodate custom circuitry. It is used in UMET-1 to interface input/output with the microprocessor. The following first describes the microprocessor side of the interface, then the real-time data and punched paper tape side of the interface.

2.4.1 The Microprocessor Side of the Interface

The Interface Board includes a switch selectable address decode which is sent to the wire-wrap counter header K1 as CS. CS is also on the header along with A0-A15 and A0-A15.
The address switches are set so that the base address of the board is $5000. Address lines 0 through 4 are cut so that the chip select (CS) pin on the board at wire-wrap header K1 pin 7, (CS is at K1 pin 8) is decoded on binary 1110 0000 000X XXXX. Using this signal and decoding uniquely address lines 0-4, each of the PIA registers can be addressed according to Table 2.4.1-1. Thus the four registers of PIA #1 are at addresses $E000 to $E003, and those of PIA #6 are at $E014 to $E017, with the other PIAs in order between. Four 7408 AND gates are used to decode A2 and A3 as shown in the table.

Each PIA has three chip select inputs; two active-high, CS0 and CS1, and one active-low, CS2. To select the PIAs at the above addresses, the selects on each PIA are wired according to Table 2.4.1-2.

To access it's four registers (two Peripheral Registers and two Control Registers) each PIA has two register select inputs, RSO and RS1. All PIAs are wired so A0 goes to RS1 and A1 goes to RSO. This appears inverted but it decreases data acquisition time by using the 16-bit data register of the 6809 to read the PIA's data port during an interrupt. The registers selected are shown in Table 2.4.1-3.

A memory map showing the addresses of all PIA registers with the format of the incoming data is shown in Figure 2.4.1-1.

2.4.2 The Real-Time Side of the Interface

The real-time input data from the TRADAT system [9] are a 100-bit serial word, transmitted at 10 words per second. The 100-bit word begins with an 8-bit synch word (11111011), followed by status indicators, spare bits, and BCD words listed in Table 2.4.2-1.
### TABLE 2.4.1-1. PIA ADDRESS LINES

<table>
<thead>
<tr>
<th>A15 – A5</th>
<th>A4</th>
<th>A3</th>
<th>A2</th>
<th>A1</th>
<th>A0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIA #1</td>
<td>Decoded by</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>PIA #2</td>
<td>switch as</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>PIA #3</td>
<td>CS ($E000)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>PIA #4</td>
<td>on the wire</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>PIA #5</td>
<td>wrap header,</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>PIA #6</td>
<td>Kl</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

### TABLE 2.4.1-2. PIA CHIP SELECT INPUTS

<table>
<thead>
<tr>
<th>PIA #</th>
<th>CSO</th>
<th>CS1</th>
<th>CS2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A4</td>
<td>A2A3</td>
<td>CS</td>
</tr>
<tr>
<td>2</td>
<td>A4</td>
<td>A2A3</td>
<td>CS</td>
</tr>
<tr>
<td>3</td>
<td>A4</td>
<td>A2A3</td>
<td>CS</td>
</tr>
<tr>
<td>4</td>
<td>A4</td>
<td>A2A3</td>
<td>CS</td>
</tr>
<tr>
<td>5</td>
<td>A4</td>
<td>A2A3</td>
<td>CS</td>
</tr>
<tr>
<td>6</td>
<td>A4</td>
<td>A2A3</td>
<td>CS</td>
</tr>
</tbody>
</table>

### TABLE 2.4.1-3. PIA REGISTER SELECTION

<table>
<thead>
<tr>
<th>A1</th>
<th>A0</th>
<th>Register Selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>PA (Peripheral Register A)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>PB (Peripheral Register B)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>CRA (Control Register A)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>CRB (Control Register B)</td>
</tr>
</tbody>
</table>
The eight-bit word in address E000 contains the two most significant decimal digits (thousands d4 and hundreds d3) of the "met" datum. E001 contains the two least significant digits (tens d2 and units d1). Notice that the most significant decimal digit of azimuth (d5) requires only two bits, that of seconds (d3), being five or less, requires only three bits. The bits representing the tens of minutes requires only three bits, one bit of which is bit 8 in the PB register of PIA 5. Addresses E016 and E007 are used for input interrupt control (see Sections 3.3.6 and 3.3.7).

<table>
<thead>
<tr>
<th>hex address</th>
<th>PIA</th>
<th>register</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>E000</td>
<td>PIA 1</td>
<td>PA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E001</td>
<td></td>
<td>PB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E002</td>
<td></td>
<td>CRA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E003</td>
<td></td>
<td>CRB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E004</td>
<td>PIA 2</td>
<td>PA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E005</td>
<td></td>
<td>PB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E006</td>
<td></td>
<td>CRA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E007</td>
<td></td>
<td>CRB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E008</td>
<td>PIA 3</td>
<td>PA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E009</td>
<td></td>
<td>PB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E00A</td>
<td></td>
<td>CRA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E00B</td>
<td></td>
<td>CRB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E00C</td>
<td>PIA 4</td>
<td>PA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E00D</td>
<td></td>
<td>PB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E00E</td>
<td></td>
<td>CRA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E00F</td>
<td></td>
<td>CRB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E010</td>
<td>PIA 5</td>
<td>PA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E011</td>
<td></td>
<td>PB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E012</td>
<td></td>
<td>CRA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E013</td>
<td></td>
<td>CRB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E014</td>
<td>PIA 6</td>
<td>PA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E015</td>
<td></td>
<td>PB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E016</td>
<td></td>
<td>CRA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E017</td>
<td></td>
<td>CRB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 2.4.1-1. PIA memory map.
<table>
<thead>
<tr>
<th>Field</th>
<th>Word Length</th>
<th>Bit Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame sync</td>
<td>8</td>
<td>1-8</td>
</tr>
<tr>
<td>Spare</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>Met flag</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Spare</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>Tracker mode</td>
<td>1+1+1+3+3</td>
<td>12-20</td>
</tr>
<tr>
<td>Time of day (hr, 2 digit BCD)</td>
<td>2+4</td>
<td>21-26</td>
</tr>
<tr>
<td></td>
<td>3+4</td>
<td>27-33</td>
</tr>
<tr>
<td></td>
<td>3+4</td>
<td>34-40</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>41-44</td>
</tr>
<tr>
<td>Elevation (deg, 4 digit BCD)</td>
<td>4+4+4+4</td>
<td>45-60</td>
</tr>
<tr>
<td>Spare</td>
<td>1+1</td>
<td>61-62</td>
</tr>
<tr>
<td>Azimuth (deg, 5 digit BCD)</td>
<td>2+4+4+4+4</td>
<td>63-80</td>
</tr>
<tr>
<td>Met word (μsec, 5 digit BCD)</td>
<td>4+4+4+4+4</td>
<td>81-100</td>
</tr>
</tbody>
</table>
The last BCD word is slant range or "met frequency" (more precisely, the period in microseconds of the audio tone transmitted from the meteorological sonde), depending on a switch set by the TRADAT operator. The switch allows the operator to choose only slant range (for non-meteorological applications of TRADAT), only met frequency; or alternating range and met data (when both are needed as in DATASONDE rocket meteorological soundings). Slant range is not provided in RAWINSONDE soundings, and five per second data rate is adequate, so it is expected that the alternating mode will be chosen universally for meteorological applications of TRADAT. In any case, the met flag (bit 10) is "1" when the 100-bit word contains met frequency, and "0" when it contains slant range. UMET-1 synchronizes on the first six bits of the synch word (111110), but also requires bit 10 to be "1". Note also that though the five "ones" in the synch word indicate "near synch" (i.e., no contiguous BCD string produces five "ones"), the added "0" is needed to assure proper synch when the (preceding) least significant bits of the last word of the 100-bit string are "ones".

The real-time serial signal enters the Interface Board from the motherboard through edge connector pin 25 and is fed to a string of thirteen shift registers as shown in Figure 2.4.2-1. Shift register #1 and #2, containing the leading bits of the 100-bit stream, is connected to the interrupt decoder. When the synch word and the "met" flag are detected, the IRQ interrupt is initiated through CA1 of PIA #1, and the content of the shift register, hardwired to the PIA's, is read into the processor. Eighty of the 100 bits are available for the processor at the PIA ports as shown in Figure 2.4.2-1. Of the 104 bits of real-time
Fig. 2.4.2-1. Real-time shift registers and interrupt logic.
data stream held by the thirteen shift registers, only the 80 (bits 21 to 100) are "ported."

2.4.3 **Punched Paper Tape Side of the Interface**

Serial data from the punched paper tape circuit on the Input Board, together with the associated clock and interrupt signals, are fed through the bus to the Interface Board. The data are clocked into the paper tape shift register. When the register is filled, the interrupt line goes active at CAI of PIA #6. Port B of PIA #6, hardwired to the paper tape shift register, contains then the data to be read by the processor.

2.5 **Read Only Memory (ROM) Board**

The ROM Board is a Motorola M68MM04A ROM/EPROM Module Micromodule 4A [10]. In order to provide two banks of program memory at the same address ($0000 to $7FFF), the board was modified as shown in Figure 2.5-1. Address line A15 has been cut and rerouted to the board select logic as follows. Address line A15 is first inverted using an unused gate in chip U40 [Reference 10, Figure 4-2], then is routed to the chip select decoder through AND gate U15, pin 10. (The input to pin 10, not shown in Figure 4.2 of Reference 10, was disconnected from +5V, and reconnected to the A15 output of U40). The ROM board is therefore enabled only when A15 is low. The control line from PIA #1, output port CB2, (through INTERFACE board pin #19) is connected directly to the bank select logic, at bus contact 33, the point from which A15 was cut.

The board is configured according to procedures described in Reference 10, so that the upper memory sockets are at $0000 to $7FFF, and the
Fig. 2.5-1. ROM Board modification for switching banks.
lower (would be) at address $8000 to $FFFF. The upper memory is selected when the PIA #1, port CB2, is low, and the lower when high. Due to the way A15 is used on the board select logic, the board is never enabled at addresses $8000 and above. To the processor, the lower memory sockets, when used, also appear at $0000 to $7FFF. The PIA input is controlled by software so that only bank A (upper sockets, the "AMET" program) is read during real-time processing, and only bank B (lower sockets, the "BMET" program) is read during the post-apogee computations of UMET-1.

2.6 Random Access Memory (RAM) Board

The RAM board is a Motorola 32K Dynamic RAM Module MEX 6832-22, [11]. The board is configured to be a 24K RAM at addresses $8000 to $DFFF, by wiring the address select wire-wrap header as shown in Figure 2.6-1. Row A is selected by decoding addresses A15 and A14. Row B is selected using A15 by way of board select, A14, and A13.

2.7 Central Processor Unit (CPU) Board

The CPU Board is a Motorola M68MM19A Monoboard Microcomputer Micro-module 19A [12], with an industry standard 2716 2k x 8 EPROM in socket U28 [Reference 12, pp. 2.2, 2.3, compare Figure 1.3-1, above]. A jumper is added between U28 pin 12 (ground) and header K9 pin 6. This grounds the 2716 chip enable(pin 18), which is an active low input. Maintaining this input active significantly decreases the data access cycle time, allowing the 2716 to operate at 2 MHz. (The long jumper between U20 pin 27 and U27 pin 21 is vestigial -- the fast chip for which it was intended became unnecessary when the 2716 proved adequate.)
Fig. 2.6-1. RAM address select header, 24k.
3.0 PROCESSOR SOFTWARE

The overall sequence of the UMET-1 program is shown in Figure 3.0-1. Recalling the general memory map (Figure 1.3-1), turning "on" the front panel switch automatically sends control to the beginning of the MONITOR program (address $F800). The UMET-1 MONITOR program, contained in the CPU EPROM, first initializes the system then sends control to address $0050 of AMET which, in turn, is contained in bank A of the ROM Board.

AMET is a set of routines which requests and receives keyboard and punched paper tape inputs before launch, and, after launch, processes radiosonde flight data in real-time. After the radiosonde reaches apogee, or when the operator stops real-time processing (by pressing the "S" key), AMET interpolates the accumulated flight data uniformly at regular intervals for final processing by BMET, and returns control to the MONITOR at address $F801.

The MONITOR then masks the real-time interrupt, clears RAM memory except COMMON, switches ROM banks, and sends control to BMET (address $150). BMET, contained in Bank B of the ROM Board, processes the interpolated data from AMET (in COMMON), and prints the final meteorological results of the radiosonde sounding. The system "idles" in an infinite loop, displaying "NORMAL PROGRAM TERMINATION," at the end of BMET until UMET-1 is turned "off."

The following first describes the interrupt system of UMET-1 (Section 3.1), MONITOR (Section 3.2), then discusses, in turn, the software routines comprising AMET (Section 3.3), and those comprising BMET (Section 3.4). The complete program of UMET-1 is listed in Appendix A.
Fig. 3.0-1. Overall sequence of the UMET-1 program.
3.1 Interrupt Design

UMET-1 is designed to process data in real-time, i.e., a real-time portion (AMET) is programmed in such a way as to allow computation within standard FORTRAN routines to be interrupted in order to read data arriving at regular intervals through PIA's. The real-time data are interpreted and stored in a queue (first-in-first-out, FIFO, file) to await processing according to the subsequent availability of the processor. The queue can hold up to one hundred fifty 0.2 second points, or thirty seconds of real-time data. Experience during development of the target system has seldom seen the queue backed up more than forty points, and then only momentarily when considerable real-time printing was required. Normally, the queue remains essentially empty.

The normal maskable interrupt (IRQ) of the 6809 processor is used, which causes the entire machine state to be stacked. Extensive use of the processor is therefore available for servicing the interrupt, without damaging the subsequently resumed computations. Except for RESET, UMET-1 uses none of the other available interrupts (NMI, FIRQ, SWI, SWI2, SWI3). If any of these unused interrupts occur, MONITOR simply issues a warning message to the terminal display "WARNING SYSTEM INTERRUPT HAS OCCURRED," and returns to the program. This warning, the Interrupt Error Message, is used throughout UMET-1 to signify any spurious interrupts.

The two interrupts used in UMET-1, RESET and IRQ are vectored as shown in Table 3.1-1. Thus UMET-1 software has four interrupt service routines: REAL, TAPE1, the MONITOR itself for RESET, and the Interrupt Error Message.
TABLE 3.1-1. INTERRUPT VECTORING

<table>
<thead>
<tr>
<th>INTERRUPT LINE</th>
<th>INTERRUPT VECTOR</th>
<th>PSEUDO VECTOR</th>
<th>ROUTINE CALLED</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESET</td>
<td>$F800</td>
<td></td>
<td>Beginning of Monitor</td>
</tr>
<tr>
<td>IRQ</td>
<td>$DFFD, A = $5878</td>
<td>$5620 $F900</td>
<td>TAPE1 REAL Interrupt Error Message</td>
</tr>
</tbody>
</table>
Upon the occurrence of a RESET or an IRQ interrupt in a 6809 microprocessor, control goes to the top of memory to read the location of the next instruction. In UMET-1, $FFFE (RESET) contains address $F800, $FFFF (IRQ) contains $DFFD, and the rest contain $F900. Thus when power is switched "on" (or the reset button on the CPU Board is depressed), control always goes to the beginning point of MONITOR, i.e., RESET always restarts the entire UMET-1 program.

When an IRQ interrupt occurs, control finds in $DFFD a jump instruction with address A, where A is determined by software as described in the next two paragraphs. Incidentally, MONITOR always masks the IRQ interrupt line (by setting the condition code register bit I = 1), so an IRQ interrupt is ignored while control is in the MONITOR.

Initially, of course, $DFFD, and its address bytes $DFFE and $DFFF, contain zeros, as does the rest of RAM. Early in the MONITOR, $DFFD is loaded with the jump instruction. Just preceding the input of punched paper tape input (the pressure calibration table of the sonde baro-switch) in AMET, MAIN calls subroutine SETUP which in turn sets A = $5878, the beginning address of TAPE1.

Interrupt service routine TAPE1 will service punched paper tape input. Later, after the completion of all paper tape input and just preceding the reading of real-time data, MAIN calls subroutine INTERR which, in turn, sets A = $5620, the beginning address of interrupt service routine REAL. REAL will service real-time data input. Finally, at the completion of AMET, MONITOR sets A = $F900, which is the error message address in MONITOR, mentioned above. The remaining UMET-1 program (BMET) requires no interrupts.
3.1.1 The Queue, Real-Time Buffer

Occasionally real-time data appear at the PIA's before the processor has completed the preceding work. A queue is used to hold up to 150 real-time points, so that each point can be read in as it occurs at the PIA's, and can be read out as rapidly as the processor can "get to" them. The processing time required between points varies, depending primarily on the amount of printout required.

Each 80-bit real-time point read from the PIA's is interpreted by the interrupt service routine REAL and stored as ten words in the queue (the IDATA array). The ten words include the time-of-day, tracking angles (azimuth and elevation), sonde frequency, etc. Pointers are defined in such a way as to create, in effect, a 1500-word loop into which data are stored ten words at a time, in sequence, and are read ten words at a time in the same sequence (first-in-first-out). If the capacity of the queue is exceeded, the oldest points are lost, but processing can continue as if the lost points were "drop outs" in the real-time data. The queue is considerably larger than necessary for normal operation, however, in fact, large enough to allow anticipated future additional printer functions.

3.1.2 Interrupts While Advancing the Queue

Subroutine ADVANC manipulates the pointers of the queue each time it accepts (the oldest point of) real-time data for processing. On the other hand, as each new real-time point arrives at the PIA's, the interrupt service routine REAL manipulates the pointers to add the point to the queue. If ADVANC is in the process of "moving" pointers in the
I queue when a real-time interrupt occurs (INTMSK = 1) UMET-1 allows REAL to read and temporarily store the new real-time point, but prohibits REAL from entering the point into the queue. If thus prohibited, REAL sets INTMSK = 2 and returns to ADVANC, ADVANC then completes its work with the queue, but seeing INTMSK = 2, then calls subroutine GETBCK to complete the work undone by REAL. In this manner, manipulation of the queue in ADVANC is always completed without interference with queue manipulation in REAL. The 0.1 second period of the real-time interrupts is sufficiently large that the next interrupt never occurs before completion of queue manipulation in REAL and GETBCK.

3.1.3 Manual Stop, STOPER

Provision is made to manually effect "apogee," the end of flight data, by depressing "S" on the keyboard. AMET, in subroutine ADVANC, calls subroutine STOPER whenever the queue is empty (IFLAG = 0). If the "S" character is detected, AMET ceases to accept real-time data (ADVANC sets MASK = 4), completes its preparation of the accumulated flight data for final processing by BMET, and returns to the MONITOR for switching to BMET.

Subroutine STOPER is also used to permit the operator to abort the reading of the punched paper tape, in favor of keyboard entry of the pressure calibration table or of re-reading the tape.

3.2 MONITOR

MONITOR is the control program for UMET-1; it begins running automatically when the power is turned on and starts the other programs in proper sequence. MONITOR is located, with the Interrupt Error Message
routine, in a separate EPROM memory on the CPU board. All the other programs are grouped in Bank A (AMET) and Bank B (BMET), the two software-switchable banks of EPROM memory on the ROM board.

As discussed in section 3.0, the AMET and BMET programs are consecutively switched into the memory map and executed. First AMET handles the real-time data phase, then BMET produces the meteorological printouts. MONITOR's main tasks are (1) initialize memory and input-output devices for AMET and BMET, including setting up the "pseudo-interrupt vector" to direct interrupts to the proper service routine, (2) do the memory bank switching, and (3) start the execution of AMET and BMET. A more detailed description of what MONITOR does follows.

MONITOR is started by a RESET interrupt initiated either by turning the power switch on, or by pressing the reset switch on the CPU board. This interrupt causes the microprocessor to get the MONITOR's starting address ($F800) from the interrupt vector location $FFFE and jump to it.

First MONITOR masks the IRQ interrupt, so spurious interrupts will be ignored until the AMET program is ready to read the paper tape. Next the ACIA is configured to communicate with the terminal. Then all RAM memory is cleared to zero. Next Bank A of EPROM memory (AMET) is switched into the memory map. This is done by writing $30 into location $E003, which is register CRB of PIA #1.

Next the "pseudo-interrupt vector" is set up. The microprocessor's permanent interrupt vectors are located in the MONITOR EPROM and cannot be changed. However, the IRQ interrupt is used by both the paper tape reader and the real-time data shift registers at different times to ask for service. So the IRQ interrupt must cause a jump to TAPE1, the paper
tape reader service routine, during paper tape read-in, but also cause a jump to REAL, the real-time data service routine, later during the sonde flight. This problem is resolved by creating a "pseudo-interrupt vector"; a jump instruction in RAM memory whose destination address can be changed during execution. An interrupt causes the processor to jump to this instruction, which in turn jumps to the proper service routine. So MONITOR stores the instruction "jump" ($7E) in location $DFFD. Before the interrupt is used, subroutines in AMET will load different destination addresses into location $DFFE, setting up the jump to direct execution to the proper service routine, TAPE1 or REAL.

The system is designed, of course, to start with the AMET program. MONITOR starts AMET by jumping to the starting address ($0050) of MAIN, the main program of AMET. AMET normally terminates by returning to the next location in the MONITOR.

When AMET finishes and returns, the MONITOR switches to Bank B (BMET) by writing $38 into location $E003. Next it clears all RAM from $A000 to SFFF. The uncleared RAM contains COMMON memory holding the input for BMET. Next, since the system will need no more input interrupts, MONITOR in effect disables subsequent IRQ interrupts. This is done by storing the starting address of the Interrupt Error Message routine ($F900) into the "pseudo-interrupt vector" location ($DFFE).

MONITOR starts BMET by jumping to $0150, the starting address of PRDMAIN, the main program in BMET. BMET does not return to MONITOR after it finishes processing, but goes into an infinite "wait" loop.
3.2.1 Interrupt Error Message

The Interrupt Error Message is an assembly language interrupt service routine that is located in the MONITOR EPROM, starting at address $F900. It simply displays the message:

**** WARNING SYSTEM INTERRUPT ERROR HAS OCCURRED *****

on the terminal and returns to the program that was interrupted.

The Interrupt Error Message is used to indicate that the microprocessor has received an erroneous interrupt, and provide a return to the interrupted program. All interrupt signals not being used by the system, FIRQ, NMI, SW1, SW2, and SW3, have the starting address of the Interrupt Error Message ($F900) stored in their interrupt vector locations, so that the message is triggered if they occur. Also, the IRQ interrupt is not used during BMET, so before BMET begins MONITOR stores the address of the message in the "pseudointerrupt vector" location $DFFE. This causes IRQ interrupts during BMET to trigger the message.

3.3 AMET

The following describes each routine of AMET, essentially in order of occurrence. Figure 3.3-1 relates the routines of both AMET and BMET in a block diagram, and indicates those coded in assembly language. Interrupt service routines TAPE1, REAL, and MONITOR are shown in the figure. MONITOR is discussed above (Section 3.2), TAPE1 in Section 3.3.4, and REAL in Section 3.3.9. The FORTRAN functions ADIR, ALOG10, FLOAT, ZSIN, and ZCOS are indicated in the figure and are discussed in
Fig. 3.3-1. Software routines of UMET-1. IRQ interrupts are masked by MONITOR until after SETUP, are then serviced by TAPE1 until INTERR, then by REAL until the end of AMET. Otherwise, all interrupts cause an Interrupt Error Message. AMET processes input in real-time, BMET computes output at the end of balloon ascent.

* The asterisk denotes assembly language routines; otherwise they are FORTRAN routines.
Section 3.4.6. Again, the program is viewed in two principal parts, the real-time or input processing part AMET, and the output processing part BMET, the latter described in Section 3.4 below. A complete listing of each routine, with comments, is included in Appendix A. Extensive reference is made to the preceding documents, [1], [2], for supplementary information concerning the FORTRAN routines of BMET and AMET respectively. Introductory information in Reference [2] is not repeated here but serves well to understand AMET.

3.3.1 MAIN

MAIN initializes AMET, requests, reads, checks, stores, displays, and prints (a) preflight values entered by the operator at the terminal, and (b) pressure calibration values entered by the paper tape reader. Required input quantities are described in Chapter 4. Obviously incorrect pressure calibration values are automatically adjusted by interpolation. The values before and after adjustment are displayed and printed to the attention of the operator and eventual user of the data. Provision is made for accepting the data, or of re-entering it by keyboard or by the punched paper tape reader. After the pre-flight information is entered and accepted, and real-time input data has commenced, MAIN displays time-to-go each second, on the terminal. After balloon release (launch), MAIN conducts the real-time processing of flight data (arriving five points per second), using subroutines ADVANCE, TRACK, and DECOM.

After the termination of real-time processing MAIN prints the decommutated data, and calls subroutine INTERP to prepare the "one-minute" table (flight data at uniform 60 second intervals) required by
BMET. Finally MAIN prints the one-minute table and certain auxiliary information such as the baroswitch contact number reached, the reason for termination (balloon burst, stopped by operator, noisy data, etc.), and calls subroutine JUMPER which returns control to MONITOR to commence BMET.

3.3.2 PRNTER

The assembly-language subroutine PRNTER initializes the printer to print a 132 character line and disables the real-time data shift register until the program is ready to accept real-time data.

When the printer is first turned on the "standard monospaced" character set is selected automatically, which prints a maximum of 80 characters per line [4]. To accommodate the wide printout which the meteorological programs generate, the "condensed monospaced" character set which prints 132 characters per line, must be selected instead. PRNTER does this by transmitting the ASCII control characters ESC, DC4 to the printer (see [4] page 1-3), using the FORTRAN input-output routines in the IOPKG [13] (see also IOPKG, Appendix A). First, PRNTER calls IOPKG subroutine INITLZ (location $5B52) [13, p.E-5]. This initializes IOPKG for use. The IOPKG subroutine LOUTCS (location $5COA) is called with the character to be sent, ESC ($1B), in the A register. This sends ESC to the printer. Then LOUTCS is called again with the character DC4 ($14) in the A register so that DC4 is sent.

The real-time shift registers must be turned off during the paper tape reading to prevent spurious interrupts. Register CRB of PIA #2 at address $E007 controls the real-time shift registers. PRNTER turns off
the shift registers by setting bits 6, 5, and 4, and clearing bit 3 of CRB [14]. The real-time shift registers stay off until they are turned on again by INTERR, at the beginning of real-time processing.

3.3.3 SETUP

The assembly-language subroutine SETUP sets up the system for paper tape reading. First, SETUP initializes the variables used by the paper tape interrupt routine TAPE1. Then, it programs peripheral register B of PIA #6 (location $E015) to receive the character from the paper tape reader. This is accomplished by storing $04 in register CRB of PIA #6 (location $E017). Next, it loads the address of routine TAPE1 ($5878) into the pseudo interrupt vector location $DFFE; subsequent interrupts from the paper tape reader will cause the processor to jump to TAPE1 for service. SETUP then programs the interrupt circuitry of PIA #6. During paper tape reading, the reader (by the presence of the sprocket hole) signals the processor when a new character is valid at PIA #6 by a positive going transition on pin CA1 of PIA #6 which causes an interrupt through the PIA's interrupt logic. The interrupt signal continues until it is cleared by a read operation on peripheral register A of the PIA (location $E014). SETUP programs the PIA's interrupt logic by storing $07 in register CRA of PIA #6 (location $E016). Finally, SETUP turns on the interrupt system by clearing the interrupt request mask bit in the processor and returns to MAIN. The system is then ready for paper tape reading.
3.3.4 TAPE1

TAPE1 is an interrupt service routine called by the paper tape reader through interrupt IRQ whenever a new character is available for input. This routine performs two tasks: first, it removes all extraneous data, control characters, etc.; second, it interprets the calibration data by combining the digits (one character or digit per row across the tape [7]) received from the reader into calibration numbers (pressure values).

The eight-hole code on the baroswitch calibration tape is according to EIA Standard RS0244, with parity bit 5, stop bit 8, and the sprocket hole between bits 3 and 4.

The data are punched in ASCII code and consists of identification numbers, line numbers, control characters, as well as the calibration numbers. The calibration numbers can be distinguished from the other data by three tests: a calibration number is always preceded by a "space" character; a calibration number's last digit is always preceded by a decimal point; and, of course, it is numeric (having a leading "0"). One exception is that no decimal point is included when the number is zero. Since zero pressures occur only following the low pressure end of the calibration table, this exception presents no difficulty.

Initially, the routine looks for five consecutive "deletes" (all holes punched) on the tape because the data on a tape are always immediately preceded by a series of deletes. After this requirement is met, the routine looks for a "space", a decimal point, or a numeric character. When a numeric is read in, it is placed on a stack and a counter.
is decremented. In this way, the counter indicates how many digits have been entered, and eventually the number of digits in the number. Each time a space is encountered the counter is reset, thus eliminating the digits currently on the stack. For a number to be accepted, a space must be read in, followed by several digits the last of which is preceded by a decimal point. Leading zeros without a decimal point are accepted after the 120th entry. It should be noted that before a digit is placed on the stack, the MSB is masked out. This removes the leading "3" of the ASCII code.

When the above conditions are met, the calibration number is on the stack with the least significant or tenths digit to be first off. As soon as the routine detects the end of a calibration number, it begins removing the digits from the stack and combining them. Each time a digit is removed, the counter is incremented.

The first digit removed from the stack (the tenths digit) is always placed in its own memory to be later combined with the rest of the number. One by one the digits are pulled off the stack, multiplied by an increasing power of ten, and added together to form the integer part of the calibration number. If the calibration number goes into the thousands, the final (most significant) digit is multiplied by ten and then by one hundred to accommodate the limitations of the machine.

Finally, a completion flag, IFLAG, is set and a counter, BYTE18, is incremented to indicate that another entry has been processed. The calibration data are now in two memory locations, one containing the integer part and the other the fractional part, and TAPE1 returns from interrupt. In MAIN, a real addition is performed on the two numbers and the result is placed in the calibration table.
TAPE1 is enabled to respond to interrupts by SETUP when it loads the starting address of TAPE1 (R5878) into the pseudo-interrupt vector location ($DFFE). It is disabled after paper tape input by INTERR which enables REAL to respond to interrupts.

3.3.5 STOPER

Subroutine STOPER is called periodically during real-time processing. It looks to see whether the "S" key on the terminal has been pressed and, if so, sets a flag.

First, it initializes flag ISTPER to zero, then it checks the status register (location $EC14) of the Asynchronous Communications Interface Adapter (ACIA) to see whether a character has arrived from the keyboard since the last time it checked. If not, it returns. If so, it reads the character from the ACIA data register (location $EC15) and checks whether it is an "S" (upper case). If not, it returns. If it is an "S", STOPER sets flag ISTPER to 1 and returns. The calling routine then checks ISTPER to see if an "S" has been hit, and if so, terminates real-time processing.

3.3.6 OFF

Subroutine OFF disables the CPU interrupt request A (IRQA) from PIA #6, to inhibit spurious interrupts from the paper tape reader head after the tape has been read. (Line 12 of Subroutine OFF should read address E016, instead of the erroneous E017. See Figure 2.4.1-1.)

3.3.7 INTERR

Subroutine INTERR sets up the system for input of real-time data. First, it saves the contents of the X register in locations BYTE17 and
BYTE18. This is because INTERR is called by the FORTRAN program MAIN, and FORTRAN uses the X register during a "call." Since INTERR uses the X register, it must temporarily save the previous contents. INTERR next temporarily masks interrupt IRQ by setting the "I" bit in the condition code register, to inhibit premature interrupts. Next, PIA #1 through PIA #5 are programmed to input data from the shift registers. PIA #1 is configured to generate interrupts on the positive edge of a "data-word ready" signal from the shift registers on pin CB1. All other interrupt inputs are disabled. This is done by writing $07 into control register CRB of PIA #1 and $04 into all other control registers of the PIAe. The X register is used to address the PIAe. Next, TFLAG is reset to 0, since it was used as a "done" flag during paper tape input, but will be used as a "queue empty" flag during real-time data input. Next, interrupts are directed to the real-time interrupt service routine REAL by loading the starting address of REAL ($5620) into the pseudo-interrupt vector location $DFFE. Next, the saved contents of the X register are restored for FORTRAN. Next, the queue pointer is initialized to the bottom address of the queue ($812E). The shift register is then turned on,* to begin processing real-time data. This is done by setting bits 5, 4, and 3 of control register CRB in PIA #2 (location $E007). Finally, the interrupt IRQ is unmasked by clearing bit "I" in the condition code.

* Software provision of this feature is included in INTERR (line 74) but is yet untested, since the associated hardware (appropriate jumper wires) have not been installed (an oversight). The shift registers are therefore always on, making it necessary to disconnect real-time data from the input ("DATA-IN" jack) until the system is ready to process real-time data (Section 4.2, step 29, and Figure 2.4.1-1).
code register, and INTERR "returns." The system is then ready to process real-time data.

3.3.8 ADVANC

Subroutine ADVANC reads real-time data from the queue, tests subroutine STOPER (if the queue is empty) to determine whether the operator has signalled the end of processing (i.e., has pressed the "S" key), performs some editing, and calls ANGLE each minute to load TIME, AZ, and EL into the "one-minute table", VL.

The flight data, TIME, FREQ, AZ, EL, at 0.2-second data rate, are processed ten points at a time. The sample of ten points, however, is advanced only five points at a time. AMET searches for and tracks signal and detects switch times between signal dwells by examining in sequence half-overlapping 2.0-second samples of raw data. At each return for more data, ADVANC moves the 2.0-second ten-point sample (TIME, FREQ, AZ, EL) ahead one second.

In addition, at each one minute after launch, except before the tracking acquisition time TGMDAQ [2], ADVANC sends the ten-point sample to subroutine ANGLE to compute the output values of the tracking angles AZ and EL. The latter two quantities are stored for subroutine INTERP in VL(2, ) and VL(3, ), along with the associated elapsed minutes from launch in VL(1, ).

As explained in Section 3.1.2 and 3.3.9, ADVANC, by setting INTMSK=1, prohibits subroutine REAL from disturbing the queue while ADVANC is reading a point, but calls subroutine GETBCK to perform the work REAL was prohibited from completing.
The flag MASK [Appendix A, ADVANC] indicates a terminating condition, e.g., elapsed time exceeds a two-hour limit (MASK=3), or the operator terminated the processing (MASK=4).

3.3.9 REAL

Interrupt service routine REAL is called by interrupt IRQ whenever a real-time word is available for input from the PIAs. It first reads the data from the PIAs into temporary storage variables BYTE1-BYTE10. It then checks the "queue busy" flag INTMSK.

If INTMSK ≠ 1, REAL reformats the real-time data to ten integers: hours, minutes, seconds, tenth-seconds, degrees of elevation, tenth-degrees of elevation, degrees of azimuth, tenth-degrees azimuth, the thousands place of the MET word, and the lower 3 digits of the MET word. REAL stores these ten integers in the queue as one "real-time point". If the queue is full (150 real-time points), the oldest real-time point is overwritten. REAL then updates the queue pointers ITOP, IBOT, TEMPX, sets IFLAG equal to the number of real-time points in the queue, and returns.

If, on the other hand, INTMSK = 1, indicating that the AMET program (subroutine ADVANC) was accessing the queue at the time the interrupt occurred, REAL cannot also access the queue without disturbing the other queue activity. In this case, REAL sets INTMSK to 2 to show there is a data word in temporary storage, and returns. After the main program is through using the queue and sees INTMSK = 2, it will call GETBCK to complete the processing of the real-time point held in temporary storage, and add it to the queue.
REAL is enabled when INTERR loads its starting address ($5620) in the pseudo-interrupt vector location ($DFFE), and is disabled when the MONITOR puts the address of the system error message into the pseudo-interrupt vector at the end of real-time data processing.

3.3.10 GETBCK

If a real-time word arrives and the associated interrupt occurs while one of the AMET subroutines is taking a data point off the queue for processing, the queue access could be interrupted. The interrupt will activate the interrupt service routine REAL. Ordinarily REAL adds the data point to the queue, but in this case, if REAL accesses the queue, it will change the queue pointers, possibly causing the interrupted queue access to "lose its place." To prevent this, AMET subroutines set INTMSK, the "queue busy" flag, to "1" while they are using the queue, and clear it to "1" afterward. Before it starts processing, REAL checks this flag, and, if it is "1", puts the data point in a temporary storage space, BYTE1-BYTE10, instead of in the queue. REAL then sets INTMSK to "2", to notify AMET that the data point is awaiting processing, and returns. After AMET has completed its queue access, it asks whether INTMSK = 2. If so, it calls GETBCK to pull the data point out of temporary storage, complete the processing left undone by REAL, and to add the point to the queue.

3.3.11 ANGLE

Subroutine ANGLE edits, condenses, and smooths input tracking angle data AZ(I), EL(I), I = 1, 2, 3,..., 10 (azimuth, elevation) [2]. It computes (when called every sixty seconds) from the ten local consecu-
tive (0.2-second) values, one value assigned at the midpoint, AZ(5), EL(5). The computed value is the mean of those points lying in the five-degree interval centered on the unit degrees mode of the ten input points. Other points, including extreme values, are therefore rejected.

The unit degrees mode is the most populated one-degree interval over the ten input points. It is found by rounding to units place the input values and counting equal rounded values. When the distribution is such that more than one unit degree interval has the highest population, the one occurring earliest in time within the 1.0-second sample is used.

3.3.12 TRACK

AMET must discern from the 0.2-second raw data, first, the switch points, i.e., the points at which the baroswitch changes contacts (channels), and second, a condensed representation of the signal transmitted while on each contact. Subroutine TRACK examines the ten-point sample FREQ(i), i = 1,2,3,...,10 to determine whether the signal is in the frequency tracking gate. If so the gate is adjusted slightly (to follow the signal), and TRACK continues by returning to ADVANC via MAIN for more data and repeating the process, and summing for the mean value of FREQ over the signal dwell, until the signal switches out of the gate.

(See Reference 2, p.41)

When the signal leaves the gate, TRACK calls SEARCH.

3.3.13 SEARCH

Subroutine SEARCH scans the full range of FREQ to find the signal (See Reference 2, p.47). SEARCH returns to ADVANC via TRACK and MAIN for more data until it finds the signal, and can reposition the gate for
TRACK. If unsuccessful, processing is terminated after 100 attempts (loss of signal for 50 seconds, $\text{MASK}=1$). Processing is terminated also when successful, if the undecommutated array exceeds 900 time points (noisy data, $\text{MASK}=2$).

3.3.14 DECOM

Subroutine DECOM determines for each condensed point, COND, its proper channel (temperature, reference, high reference, or relative humidity), and for each reference and humidity switch point, its baro-switch contact number. A detailed discussion of subroutine DECOM is given in Reference 2, p.55ff.

3.3.15 INTERP

Subroutine INTERP by interpolation constructs a table of pressure, reference frequency, temperature ordinates, and relative humidity ordinates, all uniformly at one-minute intervals for input to BMET. INTERP automatically detects balloon burst, and thereupon terminates the real-time processing automatically.

3.3.16 JUMPER

The assembly-language subroutine JUMPER is called at the normal termination of the real-time data processing. JUMPER first turns off interrupts from the shift register, to avoid disturbing the execution of BMET. This is done by clearing bits 0 and 1 of register CRB in PIA #1 (location $\$E003$). It then jumps to location $\$F830$ on the MONITOR, which sets up the system for BMET. Note that, strictly speaking, there is no return from the JUMPER subroutine call.

- 51 -
3.4 BMET

BMET is the output processing part of UMET-1, and consists of five BMET routines shown in Figure 3.3-1 above. Each is described in one of the following subsections. Detailed information is contained in Reference 1, and a complete listing of each routine with comments is included in Appendix A.

BMET is basically NASA Wallops Program No.3.0.0700 ECC-PRD, excluding the ozone data processing portions and modified to the limitations of the 6809 microprocessor FORTRAN. Variable names and extraneous parts of code, vestiges of electrochemical cell (ECC) ozone data processing, input cards, and former plot and listing functions (WODC, etc.), remain apparent in the BMET program listing. Such features should not be distracting, however, but rather may be helpful to readers familiar with the parent program.

3.4.1 PRDMAIN

The function of PRDMAIN is simply to display the facts that the program has begun, and, subsequently, completed BMET. PRDMAIN calls routine PRD which conducts all the processing of BMET. Upon return from PRD, PRDMAIN idles (in an infinite loop) until UMET-1 is turned "off" (or is restarted by pressing the RESET button on the CPU board, internally).

3.4.2 PRD

Subroutine PRD utilizes input constants entered by keyboard before balloon release, flight data computed in real-time by AMET (one-minute data, VL), and certain other quantities computed by AMET, all held in
common memory during the bank-switching from AMET to BMET. After initializing BMET, PRD computes necessary calibration constants using portions of subroutines TEMPCE and RL, checks the one-minute data for obvious errors, and proceeds to process the flight data. The basic equations and conditions used by BMET are given in Reference 1. Subroutines TEMPCE and RL are called to convert measurement data (ordinates) to temperature and relative humidity values, and to compute corresponding values of certain derived meteorological quantities (dewpoint, virtual temperature, and vapor pressure). PRD computes geopotential altitude from the measured pressure and temperature values, resolves the components of wind after calling subroutine WINDS, and computes and checks potential temperature and lapse rate. Output values are interpolated to standard pressure altitudes. The considerable amount of checking and editing done by the parent program ECC-PRD is retained in BMET (See Appendix A, PRD).

3.4.3 TEMPCE

On the first pass (ITEMP=0) subroutine TEMPCE computes from calibration inputs certain constants later used in converting temperature measurement data (ordinates) to physical units of temperature (degrees Kelvin). On each subsequent call, TEMPCE converts a temperature ordinate according to the equation given in Reference 1.

3.4.4 RL

Subroutine RL, on its first call (IH=0) computes constants needed later to convert ordinate values representing relative humidity to standard units (percent). On each subsequent call RL converts a rela-
tive humidity ordinate to percent relative humidity, but also computes the vapor pressure, dewpoint, and virtual temperature, all according to equations described in Reference 1.

3.4.5 WINDS

Subroutine WINDS computes in a single pass the wind profile speed (SPD), and direction (DIR), from the time-dependent balloon positions given by the tracking angles, AZ and EL, and altitude, HGP (computed by PRD), at the one-minute instants TIM. The algorithm is described in Reference 1.

3.4.6 Added FORTRAN Functions

The FORTRAN function ADIR is included in BMET to provide wind direction (azimuth) from the wind east-west and north-south components U and V. ALOG10 provides the logarithm to base ten of any real number, FLOAT converts an integer to a real number, and ZCOS and ZSIN provide the trigonometric functions cosine and sine for positive and negative arguments.
4.0 OPERATION OF UMET-1

The UMET-1 system consists of three units (Figure 1.0-1) most conveniently shipped or stored in cartons designed with sufficient protection against shock and damage. Properly packed, the printed circuit boards may be left in place in the card cage for shipment and storage. With reasonable handling, the system can be set up and operated with a minimum of training. The following gives a step-by-step procedure for setting up and operating UMET-1, and explains the prompts and displays presented on the video terminal during pre-and post-launch operation.

4.1 Equipment Set Up

Upon unpacking the three units of UMET-1, the printer and keyboard-video terminal should be prepared for operation according to their respective operator's manuals [4], [5]. Switch-settings for the terminal should be verified according to Table 1.2-1, and the printer should be loaded with continuous fan-fold paper.

Any packing materials inside the central unit should be removed, printed-circuit boards re-secured in their proper positions according to Table 2.1-1, and front panel switches turned off. Flat cable C3 from the front panel inside the central unit, should be connected to edge connector P2 on the Input Board (note the white painted mark keying the front side of the cable connector). Front panel connections of cable C3 include the real-time data jack ("DATA IN"), paper tape reader head, and its switch ("ON, PCAL READER").
Connect flat cable C2 (printer cable) to edge connector P4 on the CPU Board (note the slot key between contacts 7 and 9). Connect the other end of cable C2 to the edge connector at the rear of the printer, again taking note of the slot key.

Connect flat cable C1 (the keyboard-video terminal cable) to the edge connector P2 on the CPU Board (assure that connector pins (contact numbers) correspond, with contact numbers (#1, #2, #3, ...) increasing from left to right on the CPU Board). Connect the other end of cable C1 to the "D-type" connector (P3, RS-232) on the terminal.

Connect the units to 120 VAC power, and turn "ON" the power switch on each unit. The red light at the "POWER" switch should indicate that power is on, and the video screen should display the UMET-1 header:

```
UNIVERSITY OF UTAH
************* UMET-1 **********************
REAL TIME PROCESSING OF METEOROLOGICAL BALLOON SOUNDING DATA

Potentiometer R4 (Ref. Voltage Adjust) on the Input Board (Figure C-4) adjusts the threshold of the input comparator (U1). In some cases, such as when the real-time data are played from an analog instrumentation tape recorder or from other than the TRADAT system, it may be necessary to adjust R4 according to the amplitude, bias, and quality of the input voltage. Such an adjustment is easily facilitated by monitoring the input and output of comparator U1 using an oscilloscope. This adjustment, if necessary, can be made using any representative interval of the input signal at the "DATA-IN" jack.
```
Having proper output at comparator U1, ready conditions at each of the three units, and the "DATA-IN" jack disconnected, the UMET-1 is ready for keyboard inputs associated with the RAWINSONDE sounding at hand. The following describes the operating procedure at the keyboard.

4.2 Operating Procedure

1. Turn "ON" the power switch. The terminal immediately displays a UMET-1 header and requests the following keyboard inputs in turn:

2. Enter the "printer output code", normally "1". (Press the "1" key, then the "enter" key). Additional diagnostic information can be printed after or during real-time data processing by choosing "2" or "3" instead of "1", but the additional information takes considerable time and paper, and is useful only to one who is intimately familiar with the processing algorithm. After the operator enters his choice, the printer prints a UMET-1 header and labels the next following printout "INPUT DATA".

3. Enter station ID ("72" for NASA Wallops Flight Center). (Station IDs and geopotential heights are listed in Reference 1, Table 1).

4. Enter station geopotential height in meters ("4.0" for NASA Wallops Flight Center at Meteorological Station, Bldg. X-85).

5. Enter zero azimuth, "0" for north, "1" for south. This is determined by the tracking system used.

6. Enter launch month of the year, integer "1" to "12".
7. Enter launch day of the month, integer "1" to "31".
8. Enter launch year, two digit integer, e.g., "83".
9. Enter the meteorological temperature at the station (launch site), (degrees centigrade).
10. Enter the local relative humidity at the station (%).
11. Enter the atmospheric pressure at the station (millibars).
12. Enter the local wind speed at the station (meters per second).
13. Enter the local wind direction at the station (degrees).
14. Enter sonde ID, limited to an integer less than 32000.
15. Enter "unadjusted reference ordinate", less than 100, to nearest tenth. This is the ordinate corresponding to the reference frequency at launch, when the recorder scale factor (reference adjustment) is such that a 60 hertz input corresponds to an ordinate of 30 (see [2], p.11). This value is determined by measurements of sonde output before balloon release.
16. Enter air temperature calibration (ordinate value corresponding to 30 degrees centigrade).
17. Enter relative humidity calibration (ordinate value corresponding to -40 degrees C and 46 ordinates).
18. Enter air temperature ordinates at balloon release.
19. Enter relative humidity ordinates at balloon release.
20. Enter sonde type, normally "2" for the standard National Weather Service sensors [temperature element ML-419, relative humidity element ML-476 (carbon)].
21. Enter baroswitch contact numbers at burst, if known (as when processing recorded data). No entry defaults to the maximum value, 180. The printer then lists all of the above inputs.

22. Enter the choice: "1" if the baroswitch pressure calibration table will be entered by punched paper tape, or "2" if by keyboard.

23. Enter the PCAL table accordingly. If by punched paper tape: place leading end of the tape in the reader head, properly seat the tape and close the head, position the tape with at least five rows of "all holes punched" (DELETES) preceding the punched data. Turn "ON" the tape reader, and pull the tape through from left to right at any speed but without reversing the direction. After the tape is pulled completely through, turn "OFF" the tape reader. The printer tabulates the pressure calibration values read from the punched paper tape, the "effective" baroswitch contact number at launch (to the nearest one-hundredth contact position), and the highest contact number calibrated. The terminal then requests acceptance or rejection of the pressure calibration table.

24. Enter acceptance, "1", or rejection, "2", of the PCAL table. If rejected the terminal requests repeat from step 23 above.

25. Enter the hour of launch, two-digit integer "1" to "23", Greenwich Meridian Time.

26. Enter minutes of launch time, two-digit integer "1" to "59".

27. Enter seconds of launch time, to the nearest tenth second. The terminal offers the opportunity to re-enter the launch time in case of launch delays.
28. Enter rejection ("2") of the entered launch time and repeat from step 25 above (this allows last minute changes in the launch time), or enter acceptance ("1") so UMET-1 can begin accepting input data, reading the time word, "counting down" to launch time, and, at launch, to begin to process the input data. If a launch abort occurs after acceptance of the entered launch time, UMET-1 must be restarted (turned "OFF", then "ON") to initiate a new sounding.

29. After acceptance of the entered launch time, plug in (at "DATA IN") input data from the TRADAT or magnetic tape playback system. No further action is necessary on the part of the operator until balloon apogee. (Reliable operation requires that no real-time data be applied to the DATA-IN jack until acceptance of launch time in step #28 above. Software provision to automatically inhibit premature interrupts from the real-time circuit, not completed at the termination of the project, is a recommended future improvement.)

30. Enter "S" key (upper case, shift, or alpha-lock S) to terminate real-time data processing. This normally should occur at apogee (balloon burst), but may be used for any reason.

31. Turn "OFF" the power switch after the terminal displays "NORMAL PROGRAM TERMINATION".

4.3 Description of Video Display Output

After acceptance of the launch time entered by keyboard (4.2 step 28), the terminal displays "*** SET UP TO RECEIVE REAL TIME DATA ***,"
and real-time data is plugged in at the "DATA-IN" jack, the system
starts reading appropriate biphase signal (see Section 2.3.1). As the
time word in the data approaches balloon release time (entered in steps
25, 26, 27, Section 4.2 above), the terminal displays the time-to-go
(negative values of elapsed time from launch) each second. After launch
time, the terminal displays the condensed points [2] as they are com-
puted. The index, elapsed time (time of occurrence in seconds after
launch), dwell time (duration of the condensed point in seconds), and
the mean frequency of the condensed point are displayed. The progress
of the processing is conveniently monitored by observing the condensed
points. The operator also may verify balloon burst and discern other
facts concerning sond performance.

After each reference point is processed, the terminal displays the
progress of the automatic adjustment of the reference frequency thresh-
holds RFL and PFL [2]. Sonde frequencies above RFL are identified as
reference signal — those above PFL are distinguished as high refer-
ence. Since sondes differ in reference frequency, and drift in fre-
quency during flight, UMET automatically adjusts RFL and PFL by "track-
ing" the sonde reference frequency. The display periodically shows the
received COND3 (current reference frequency), and the computed RFL and
PFL.

Occasionally, the balloonsonde "ices" and temporarily reverses its
ascent. An AMET (DECOM) algorithm treats such special cases but also
notes in the output the time and contact number at which each dip
occurred.
At termination of real-time processing, the terminal rapidly displays intermediate information (decommutation parameters for each condensed point) which can be ignored under normal operation. This diagnostic information is printed, if desired, under printer options 2 and 3 (4.2, step 2). The display indicates "REAL TIME PROCESSING COMPLETE," and the terminating condition, e.g., "STOPPED BY THE OPERATOR."

Successful bank switching from AMET to BMET (Section 3.0) is signaled by the terminal display: "AT THE START OF PRD." After a few minutes of processing by BMET and subsequent printing of BMET output, the terminal displays "NORMAL PROGRAM TERMINATION."
5.0 CONCLUSIONS AND RECOMMENDATIONS

UMET-1 demonstrates the utility and practicality of a rather powerful, portable, on-site, real-time data processing capability for meteorological soundings. Additional effort, of course, would both improve the existing system and extend its applicability. Examples for improvement include the addition of the recommended real-time plotting feature (see Section 1.1), the installation of connectors at the rear panel of the central unit for connecting cables to the printer and keyboard terminal, and a more secure placement of the input cable C3 inside the front panel of the central unit. Correcting the address error in Subroutine OFF (Section 3.3.6) would eliminate the necessity of turning off the paper tape reader head (Section 4.2, step 23) after use, and completing the implementation of software control of the real-time input shift registers (Section 3.3.7) would eliminate the necessity of leaving the "DATA-IN" jack disconnected during the preflight input procedure. Further operational experience with the system will suggest additional adjustments both in hardware and software.

Considerable optimization of the software is recommended in subsequent versions of UMET. Existing programs RAWINPROC and ECC-PRD were adapted for UMET-1, deliberately with a minimum of software redesign. As a result, though familiar relative to the earlier processors, the system makes little use of advantages attending local, automated real-time operation. Reference to "ordinates" both in the input and output processes of AMET, and the "one-minute" condensation of data for BMET, are examples of unnecessary vestiges of antiquated systems. The entire preflight procedure can be considerably simplified.
Optimization nevertheless should retain consistency with the preceding systems, or at least clearly address the impact on meteorological statistics drawn from the new versus old processing systems. Of course, an improved system should maintain adequate backup and monitoring provisions to assure confidence and reliability. Real-time data should be recorded routinely to facilitate post-flight playback and processing. Automatic processing should accommodate available human assistance through occasional "trouble spots" in the data, particularly during reprocessing when "foresight" is provided from preceding passes of the flight data through the processor.

The UMET concept applies to other meteorological sounding systems, including rocket meteorological systems. Though initial programming for the DATASONDE has been done (Appendix B), field-recorded flight data are needed to complete the design of the associated firmware.

Redesign to rechargeable battery power would increase both utility for remote operations, and immunity to power irregularities in general.
REFERENCES


- 65 -
Complete listings of MONITOR, AMET, and BMET are given below. The subroutines of each are in alphabetic order. Also included in AMET is COMMON, and in BMET is PRDCOM, the common memory between AMET and BMET.
UMET-1 MONITOR AUGUST 1982

00001 OPT L
00002 ORG $F800
00003 F800 1A FF
00004 F802 86 81
00005 F804 1F 8B
00006 F806 2
00007 F807 12
00008 F808 12
00009 F809 12
00010 F80A 86 03
00011 F80C B7 EC14
00012 F80F 86 51
00013 F811 B7 EC14
00014 F814 86 00
00015 F816 1C FB
00016 F818 8E 8000
00017 F81B A7 80
00018 F81D 8C DFFF
00019 F820 26 F9
00020 F822 86 31
00021 F824 B7 E002
00022 F827 86 7E
00023 F829 B7 DFFD
00024 F82C OC FB
00025 F82E 7E 0050
00026 F831 0C FB
00027 F833 36 38
00028 F835 B7 E003
00029 F838 8E A000
00030 F83B 86 00
00031 F83D A7 80
00032 F83F 8C E000
00033 F842 26 F9
00034 F844 1C FB
00035 F846 86 7E
00036 F848 B7 DFFD
00037 F84B CC F900
00038 F84E FD DFFE
00039 F851 7E 0150
00040 F900
00041 F900 1C FB
00042 F902 8E FA00
00043 F905 E6 80 DOG
00044 F907 B6 EC14 PRINT1
00045 F90A 84 92 ANDA #80
00046 F90C 27 F9
00047 F90E F7 EC15

OPT L; enable IRQ interrupt
ORCC #$FFF; disable IRQ interrupt
LDA #$81; load direct page
TFR A,DP; register with $81 (?)
NOP
NOP
NOP
LDA #$03; reset the ACIA
STA $EC14
LDA #$51; configure ACIA
STA $EC14; for 7 bits, no parity
LDA #$00; zero all RAM
ANDCC #$FB; (locations
LDX #$8; $8000-$DFFF)
STA ,X+
CMPX #$DFFF
BNE AGAIN
LDA #$30
STA $E003; select Bank A (AMET)
LDX #$7E; set up "pseudo-interrupt
STA $DFFD; vector" jump
ALX ;X+
ANDCC #$FB
JMP $0050; start AMET
ANDCC #$FB
LDA #$E8
STA $EO03; select Bank B (BMET)
LDX #$A000; zero RAM
LDA #$00; from $A000 to $DFFF
STA ,X+
CMPX #$EO00
BNE BOZO
LDA #$7E; cause IRQ interrupts
STA $DFFD; to trigger
LDD #$F900; System Error Message
STD $DFFE
JMP $0150; start BMET
ORG $F900; SYSTEM ERROR MESSAGE
ROUTINE
ANDCC #$FB
LDX #$FA00; X reg = loc. of 1st
char.
LDB ,X+
GET a character, incr. X
LDA $EC14; loop until terminal
ANDA #$02; is free.
BEQ PRINT1
STB $EC15; send char. to terminal
ANDCC #$FB
CMPX #$FA37 ;reached end of message?
BEQ BABOON ;if so, go to return (BABOON)
JMP DOG ;if not, get next char.(DOG)
NOP
RTI ;return from interrupt

ORG $FFF2 ;interrupt vectors:
ORG $FFF2 ;interrupt vectors:

$F900 ;SW3, to System Error Msg.
$F900 ;SW3, to System Error Msg.
$F900 ;SW2, to System Error Msg.
$F900 ;SW2, to System Error Msg.
$F900 ;FIRQ, to System Error Msg.
$F900 ;FIRQ, to System Error Msg.
$DFFD ;IRQ, to pseudo-interrupt vector
$DFFD ;IRQ, to pseudo-interrupt vector
$F900 ;SW1, to System Error Msg.
$F900 ;SW1, to System Error Msg.
$F800 ;RESET, to Monitor
$F800 ;RESET, to Monitor
SUBROUTINE ADVANCE (JJ, TSTOP, TLANCH, TMDAO, MASK)

COMMON BLOCK

INCLUDE 'COMMON.SA'

ADVANCE, 5 NEW RAW DATA POINTS.

MASH=0

DO 2 JI=1,5
   JS=JI+5
   TIME(JI)=TIME(JS)
   FREQ(JI)=FREQ(JS)
   AZ(JI)=AZ(JS)
   EL(JI)=EL(JS)
2 CONTINUE

HANDLE REAL TIME DATA

IF(IFLAG .GT. 0)GOTO 3224
CALL STOPER
IF(I5TPER .NE. 1)GOTO 3333 ;IF STOP IS HIT ISTEPR=1
MASK=4
RETURN

INTMSK=1
IF(IBOT .GE. 150)IBOT=0 ;IBOT POINTS TO OLDEST NOT YET PROCESSED POI
IDUM=IBOT+1
IFLAG=IFLAG-1
DUMMY=IDATA(IDUM+9)
DUMMY=IDATA(IDUM+10)
FREQ(J)=(DUMMY*1000)+DUMMY1
DUMMY=IDATA(IDUM+6)
DUMMY=IDATA(IDUM+7)
DUMMY=IDATA(IDUM+1)
DUMMY=IDATA(IDUM+2)
DUMMY=IDATA(IDUM+3)
DUMMY=IDATA(IDUM+4)
SSECS=DUMMY1+(DUMMY*1)
DUMMY=IDATA(IDUM+10)
AHEU=(DUMMY*60)+SECS
HOURS=(IDATA(IDUM+1))*3600.
TIME(J)=HOURS+AMINU-TLANCH ;TIME IS IN SECONDS FROM LAUNCH
IF(INTMSK .EQ. 1)GOTO 944
CALL DETECK
INTMSK=0
IF(TIME(J) .LT. TIME(J-1)) GOTO 5
DUMMY=TIME(J-1)+60.
IF(TIME(J) .GT. DUMMY) GOTO 5

REAL TIME DATA NOW IN PROGRAM

IF(EL(J) .GE. 90.0) EL(J)=1
0059 IF(EL(J) .LT. 0.1) EL(J) = .1
0060 IF(AZ(J) .GE. 360.0) AZ(J) = .1
0061 IF(AZ(J) .LT. 0.1) AZ(J) = .1
0062 IF(FREQ(J) .LT. 1.0) GOTO 5 ; REAL DATA IS SOMETIMES 0.0
0063 FREQ(J) = 1/(FREQ(J)x.000001) ; DATA COMES IN IN USEC
0064 *
0065 IF(FREQ(J) .LT. 4.8 .OR. FREG(J) .GT. 200.) GOTO 5
0066 GO TO 3
0067 J=J-1
0068 3 CONTINUE
0069 IF(TIME(9) .GT. TSTOP) GOTO 100
0070 *
0071 *
0072 *
0073 51 IF(TIME(9) .LT. VL(1, LIST)) GOTO 53
0074 IF(TIME(4) .LT. TSTOP) GOTO 52
0075 ****************************************** CALL ANGLE ******************************************
0076 CALL ANGLE
0077 VL(2, LIST) = AZ(5)
0078 VL(3, LIST) = EL(5)
0079 52 LIST = LIST + 1
0080 IF(LIST .GE. 120) GOTO 128 ; VL TABLE IS FULL
0081 VL(1, LIST) = VL(1, LIST-1) + DLIST
0082 GO TO 51
0083 53 CONTINUE
0084 RETURN
0085 *
0086 82 CONTINUE
0087 C
0088 C END OF DATA
0089 C
0090 C
0091 RETURN
0092 C
0093 100 MASK = 1
0094 RETURN
0095 C
0096 128 MASK = 3
0097 RETURN
0098 END
0099
SUBROUTINE ANGLE

*COMMON BLOCK*************************************************************************

INCLUDE 'COMMON.SA'

DIMENSION NIAZ(10), NIEL(10), IAZ(10), IEL(10)

INITIALIZE AND QUANTIZE

DO 11 K=1,10
   NIAZ(K)=0
   NIEL(K)=0
   IAZ(K)=AZ(K)+0.5
   IEL(K)=EL(K)+0.5
11 CONTINUE

COUNT FOR DISTRIBUTION

DO 10 K=1,9
   LL=11-K
   DO 12 L=1,LL
      IF(IAZ(LL) .EQ. IAZ(L)) NIAZ(LL)=NIAZ(LL)+1
      IF(IEL(LL) .EQ. IEL(L)) NIEL(LL)=NIEL(LL)+1
   12 CONTINUE
10 CONTINUE

DETERMINE MODE INTERVAL

KMA=1
KME=1
DO 13 H=2,10
   IF(NIAZ(K) .GT. NIAZ(KMA)) KMA=K
   IF(NIEL(K) .GT. NIEL(KME)) KME=K
13 CONTINUE

COMPUTE MEAN IN MODE INTERVAL

SUMA=0.
NSUMA=0
DO 14 K=1,10
   IDUM=IAZ(K)-IAZ(KMA)
   IDUM1=ABS(IDUM)
   IF(IDUM1 .GT. 2) GOTO 14
   SUMA=SUMA+AZ(K)
14 CONTINUE
   SUME=0.
   NSUME=0
   DO 15 K=1,10
      IF(IEL(K) .NE. IEL(KME)) GO TO 15
      SUME=SUME+EL(K)
15 CONTINUE

OUTPUT
DUMMY = NSUMA
DUMMY1 = NSUME
AZ(5) = SUMA/DUMMY
EL(5) = SUME/DUMMY1
RETURN
END
COMMON IFLAG, JUNK(7), JSTPER, JXXX, INTMSK, IRAM(10), ITOP, ITOT; FOR REAL TIME
COMMON IDATA(1500), VL(7, 120), SECS, DUMMA, DUMMI, IDUMM, IDUMM1, AIMU
COMMON IDSTAT, IDSOND, IDAY, IYEAR, HEIGHT, FF0, FTEMP0
COMMON FR0, RECTP, CALRH, ICROM, CBST, NCDS, IOUT, VSFC, DSFC; PASS TO PRD
COMMON SRT, SURH, IAU, II, I2, I63; PASS DATA TO PRD
COMMON LIST, DLIST, SIGMAX, SIGMIN, HGATE, IN, SIGLEV, NSUM, FSUM
COMMON TERST, IJOIN, ITYPE
COMMON AZ(10), EL(10), TIME(10), FREQ(10), V2(7)
COMMON ICOND(2, 900), COND(3, 900), TMAN(4), ICMAN(4)
COMMON PCAL(180), TEST(10)
SUBROUTINE DECOM

SUBROUTINE DECOM(JK, TNCH, DSL, FHUM, ICR1, RFL, PFL, IDC)

DIMENSION DWELT(10), TDWELT(10), SDWELT(10)

C
C INITIALIZE DECOM
C
IF(JK .GT. I)GOTO 5
DRFPL=2.0
KFSUM=0.0
RTSUM=0.0
TR=0.0
AMLT=100.0
MLT=100
GTEMP=4.0
TSL=0.0
ESL=0.0
TR1=0.0
INCH=0
JKR=0
NOH=0
JKP=0
ICM=0
JKRI=0
M1=1
ICR=0
KROSS=0
NXTP=0

CONTINUE
IF(NXTP .EQ. 0)NXTP = 30*MLT
REJECT SHORT DWELLS
IF(COND(2,JK) .GT. 1.9)GO TO 8
ICOND(1,JK) = 8
GO TO 900

T = COND(1,JK)
Dwell = COND(2,JK)

;ELAPSED TIME TO LEADING EDGE
;DWELL TIME
A process is defined by the following steps:

1. Initialize variables and conditions:
   - \( D = \text{COND}(3, \text{JK}) \) if \( \text{VALUE}(\text{FRED}) \) is true.
   - \( T_E = T + \text{DWELL}/2 \).
   - If \( D \leq R_F \), go to 200.

2. Execute actions based on conditions:
   - If TVRST > S_E_20, go to 10.
   - If \( M_1 < 0 \), go to 179.
   - \( ICR_1 = ICR \).
   - \( SLOPE_1 = SLOPE \).
   - \( JKR_1 = JKR \).
   - \( TR = TR \).
   - \( M_1 = 1 \).
   - \( KROSS = 0 \).
   - Go to 78.

3. Forward-assign between references:
   - If \( T_{INCH} \neq 2 \), go to 78.
   - If \( JKR \neq 0 \), go to 782.
   - Go to 600.

4. Temperature gate:
   - Continue.
   - \( TF = DSL + ESL \times (T_E - TSL) \).
   - \( \text{DUMMY} = D - TF \).
   - If \( (\text{DUMMY} \geq 0) \), go to 2200.
   - \( \text{DUMMY} = \text{DUMMY} \times (-1) \).
   - If \( (\text{DUMMY} \geq 0) \), go to 783.

5. A temperature datum:
   - \( ESL_1 = ESL \).
   - \( ESL = 0.8 \times ESL + 0.2 \times (D - DSL) / (TB - TSL) \).
   - \( DUMMY = ESL - ESL_1 \).
   - \( IF(DUMMY \geq 0) \), go to 2210.
   - \( DUMMY = \text{DUMMY} \times (-1) \).
   - \( IF(DUMMY \geq 0) \), go to 2210.
   - \( DSL = 0.8 \times DSL + 0.2 \times D \).
   - \( TSL = TB \).
   - \( \text{ICOND}(1, \text{JK}) = 1 \).
   - \( INCH = 1 \).
   - Go to 900.

6. Adjust dwell successfully:
   - If \( \text{NOH} \neq 1 \), go to 784.
   - If \( JKR \neq 0 \), go to 785.

7. Restore temperature frequency gate.
PROCESS THIS NON-REFERENCE, NON-TEMPERATURE

FIRST CONTACT SWITCH POINT

IDUMMY = ICR1/MLT
ICM = (IDUMMY + 1) * MLT
DUMMY = IDUMMY
SLOP2 = AMLT * T/IDUMMY
GO TO 767

IDC = (T - T2) / SLOP2 + .5
IDC = IDC * MLT
IF (IDC < MLT) GO TO 789
IDUMMY = ICR1/(5 * MLT)
IDUMMY1 = IDUMMY * 5 + 1
IF (ICM + IDC < IDUMMY1 * MLT) GO TO 7861
ICOND(1, JK) = 6
KROSS = 1
INCH = 5
GO TO 900

CONTINUE

ICM = ICM + IDC
SLOP2 = AMLT * (T - T2) / IDC
ICOND(2, JK) = ICM / MLT
T2 = T
AN HUMIDITY DATUM

ICOND(1, JK) = 4
INCH = 4
FHUM = 0
GO TO 900

EARLY CONTACT, POSSIBLE BALLOON DIP

MI = 0
ICOND(1, JK) = 5
INCH = 5
GO TO 900
0175  C
0176  *  
0177  C
0178  C
0179  (*  
0180  600  COND(3,JY.R) = RFSUM/RTSUM 
0181  COND(2,JY.R) = RFSUM 
0182  TR = COND(1,JY.R) 
0183  IF(ICR .NE. 0)GO TO 610 
0184  C
0185  *  
0186  ICRE = TR/90. + .5 
0187  ICRB1 = ICR1/(5*MLT)  
0188  IF(ICRB .EQ. 0)ICRB=1 
0189  ICR=(ICR1+ICRB)*5*MLT 
0190  IDUMMY=ICR-ICR1 
0191  DUMMY = IDUMMY 
0192  SLOPE = AMIT/ICR/DUMMY 
0193  ICR1 = ICR 
0194  SLOPE1 = SLOPE 
0195  M1 = 0 
0196  JNSTRT = 0 
0197  GO TO 644 
0198  
0199  610  M = 1 
0200  MN = 0 
0201  620  IF(M .GT. 1) M1 = M 
0202  MN = MN + 1 
0203  ICR = ICR1 + M*MLT 
0204  IF (ICR1 .LT. (135 -(M - 1)*5)*MLT) ICR=ICR1+5*MLT*M 
0205  IDUMMY = ICR-ICR1 
0206  DUMMY = IDUMMY 
0207  SLOPE = AMIT/(TR-TR1)/DUMMY 
0208  M = SLOPE1/SLOPE1 + 0.5 
0209  IF(M .GT. 1)GO TO 620 
0210  IF(M .EQ. 1)GO TO 644 
0211  C
0212  M = 0 INTERPRETED AS BALLOON DIP, BARO'SNITCH REVERSAL, 
0213  SUCH AN EARLY REFERENCE IS IGNORED FOR PRESSURE 
0214  
0215  IDUMMY = ICR/MLT 
0216  DUMMY = TR/60 
0217  WRITE(IOUT,944)IDUMMY,DUMMY 
0218  944  FORMAT(1.X,'CHECK FOR BALLOON DIP AFTER CONTACT ',I3,
0219  ' NEAR ',F5.1,' MINUTES.') 
0220  CONTINUE 
0221  DUMMY = SLOPE/SLOPE1-1 
0222  IF(DUMMY .GT. 0.0) GOTO 2220 ;TAKE ABSOLUTE VALUE OF DUMMY 
0223  DUMMY = DUMMY*(-1) 
0224  M = MN + 1 
0225  644  CONTINUE 
0226  DUMMY = SLOPE/SLOPE1-1 
0227  IF(DUMMY .LT. 0.0) GOTO 2220 ;TAKE ABSOLUTE VALUE OF DUMMY 
0228  DUMMY = DUMMY*(-1) 
0229  MN = MN - 1 AND. DUMMY < LT. 0.3)MN=-1 
0230  IF(MN .NE. 0 .OR. M1 .NE. 0)GO TO 645 
0231  M1 = 0 
0232  DUMMY = TR1/60.0
DUMMY1 = TR/60.0
WRITE(IOUT,6450) DUMMY,DUMMY1
6450 FORMAT(' EARLY CONTACT FOUND IS FALSE BECAUSE NO',
' BALLOON DIP BETWEEN ',F6.2,' AND ',F6.2,' MINUTES.
', ' WILL BACK-ASSIGN!')

ICOND(2,JKR) = ICR/MLT ;645
ICM = ICR

GTW = 0.1 = SLOPE
HUMG = 0.25 * SLOPE

SET 'NO HUMIDITY' FLAG ABOVE CONTACT NUMBER 135

GTSW = 0.1
SLOPE

HUMGT = 0.25 s SLOPE

SET 'NO HUMIDITY' FLAG ABOVE CONTACT NUMBER 135

GTSW = 0.1
SLOPE

IF(NOH .EQ. 1)GO TO 50
IF (ICR .LT. 135-MLT) GO TO 120
NOH=1
TNOH = TR
GETMP = 6.

ADVANCE DWELT ARRAY, AND TEST FOR BURST

50 IF (TR (.LT. 3000.) GO TO 55
DO 52 IS = 1,9
IDUMMY = 10-IS
IDUMMY = 11-IS

DWELT(IDUMMY) = DWELT(IDUMMI)
DUMMI = 10-IS
SDWELT(IDUMMY) = SDWELT(IDUMMI)
52 DO 51 ISS = 1,3

DWELT(1) = SLOPE
DUMMI = 10-IS
51 SDWELT(1) = SDWELT(1) + DWELT(ISS)

SDWELT(1) = SDWELT(1) / 4.0

IF(DWELT(IS) .GT. 15.0 .AND. SDWELT(IS) .LT. SDWELT(IS+1) .AND.
SDWELT(IS+3) .GT. 70.0) IBRST = IBRST + 1

IF(ISRST .LT. 3 .OR. TR .LT. 1000.) GO TO 55

IBRST = 0
IBRST = 1.3

BURST CONDITIONS ENCOUNTERED

TBAST = TR + (CBRST-DUMMY)*SLOPE

IF (TBAST .LT. 3 .OR. TR .LT. 4000.) GO TO 55

IBRST = 0

IF (TBAST .LT. 3 .OR. TR .LT. 4000.) GO TO 55

TBAST = TDWELT(4)

WRITE(IOUT,950) IEB; IEB;
950 FORMA'T( IX,10F12.1)

IDUMMY = CBRST
TRUNCATES IE XX.X=XX

IF (ICR .LT. IDUMMY*MLT) Go TO 54

DUMMY=IDUMMY
TERST = TR + (CBRST-DUMMY)*SLOPE

GO TO 56

54 CONTINUE

IBRST = 0

IF (TBAST .LT. 3 .OR. TR .LT. 4000.) GO TO 55

IBRST = 0

IF (TBAST .LT. 3 .OR. TR .LT. 4000.) GO TO 55

TBAST = TDWELT(4)

WRITE(IOUT,950) IEB; IEB;
950 FORMA'T( IX,10F12.1)

IDUMMY = CBRST
TRUNCATES IE XX.X=XX

IF (ICR .LT. IDUMMY*MLT) Go TO 54

DUMMY=IDUMMY
TERST = TR + (CBRST-DUMMY)*SLOPE

GO TO 56

54 CONTINUE

IBRST = 0

IF (TBAST .LT. 3 .OR. TR .LT. 4000.) GO TO 55

TBAST = TDWELT(4)

WRITE(IOUT,950) IEB; IEB;
950 FORMA'T( IX,10F12.1)

IDUMMY = CBRST
TRUNCATES IE XX.X=XX

IF (ICR .LT. IDUMMY*MLT) Go TO 54

DUMMY=IDUMMY
TERST = TR + (CBRST-DUMMY)*SLOPE

GO TO 56

54 CONTINUE

IBRST = 0

IF (TBAST .LT. 3 .OR. TR .LT. 4000.) GO TO 55

TBAST = TDWELT(4)

WRITE(IOUT,950) IEB; IEB;
950 FORMA'T( IX,10F12.1)

IDUMMY = CBRST
TRUNCATES IE XX.X=XX

IF (ICR .LT. IDUMMY*MLT) Go TO 54

DUMMY=IDUMMY
TERST = TR + (CBRST-DUMMY)*SLOPE

GO TO 56

54 CONTINUE

IBRST = 0

IF (TBAST .LT. 3 .OR. TR .LT. 4000.) GO TO 55

TBAST = TDWELT(4)

WRITE(IOUT,950) IEB; IEB;
950 FORMA'T( IX,10F12.1)

IDUMMY = CBRST
TRUNCATES IE XX.X=XX

IF (ICR .LT. IDUMMY*MLT) Go TO 54

DUMMY=IDUMMY
TERST = TR + (CBRST-DUMMY)*SLOPE

GO TO 56

54 CONTINUE

IBRST = 0

IF (TBAST .LT. 3 .OR. TR .LT. 4000.) GO TO 55

TBAST = TDWELT(4)

WRITE(IOUT,950) IEB; IEB;
950 FORMA'T( IX,10F12.1)
------ ADJUST REFERENCE THRESHOLDS

GO TO 131

IF(JKP .EQ. 0)GO TO 131.

CORRECT NXTP WHEN APPARENT THAT A HIGH REF. (P) PT. WAS MISSED

IF(NXTP .GT. 135*MLT) TP12 = 15*SLOPE
IF (NXTP .GT. 135*MLT) TP12 = 5*SLOPE
IF (NXTP .GT. 135*MLT) AND. (T-TP11.GT. (1.5*TP12).AND.
MOD(ICK, IDUMMY).EQ. 0 .AND. ICR.LE.135*MLT)NXTP=ICK
DUMMY=TP1-I*TP12
IF (DUMMY .GT. 0.0) GOTO 2230
DUMMY =DUMMY*(-1)
2230 IF(NXTP .LT. 135*MLT .AND. DUMMY .LT. .5*TP12 .AND.
ICR .NE. NXTP) NXTP = NXTP + 15*MLT
IDUMMY = S*MLT
IF (NXTP .GT. 135*MLT .AND. (T-TP11) .GT. (1.0*TP12) .AND. MOD(
ICR, IDUMMY).EQ. 0 .AND. ICR.GT. NXTP) NXTP = ICR
210 IF (NXTP .GT. 135*MLT .AND. DUMMY .LT. .5*TP12
121 ICR .NE. NXTP) NXTP = NXTP + 5*MLT
131 IF (ICK .EQ. 0) GOTO 132
WRITE(IOUT,1131) ICR, NXTP
1131 FORMAT(’ CONTACT NUMBER ’,I3,’ NOT EQUAL TO NXTP ’,I3)

BACK-ASSIGN

DECOMMUTATE TEMPERATURE AND HUMIDITY UP TO THE CURRENT REFERENCE.

ASSURE DWELL PRECEDING THIS REFERENCE IS A TEMPERATURE

CONTINUE

IF(MDH .EQ. 1)GO TO 179
IF(COND(1,JKR-1) .LT. TR = .0*SLOPE)GO TO 181
IF(COND(1,JKR-1) .EQ. 1)GO TO 181
COND(1,JKR-1) = 10 + ICOND(1,JKR-1)
ESL = 0.
TSL = COND(1,JKR-1) + COND(2,JKR-1)/2.
DSL = COND(3,JKR-1)
KROSS = 1
181 IF (KROSS .NE. 1) GOTO 188
JNFIN = JKR-2
INCHN = 2
ESLN = ESL

PAGE 06 DECOM SA:1 UMET-1 15 DEC 1982
CENTRAL PROGS
OF FGUR QUALITY

PAGE 007	DECOM	. SA:1	UMET-1	15 DECEMBER 1982

0349 1SH = TSL
0350 DSLN = DSL
0351 NDUM2 = JNFIN - JNSTRAT + 1
0352 GO 1870 JN1 = 1, IDUM2
0353 JN = JNFIN + 1 - JN1
0354 *
0355 C SKIP REFERENCE AND REJECT POINTS
0356 *
0357 MM = 90
0358 IF (M1 .EQ. 1) GO TO 182
0359 MM = 80
0360 GO TO 186
0361 182 TN = COND(1, JN)
0362 DWELLN = COND(2, JN)
0363 DN = COND(3, JN)
0364 TN = TN + DWELLN/2.
0365 TS = (TN - TR1)/SLOPE
0366 NCT = TS + .5
0367 DUMMY = NCT
0368 TO = DUMMY*SLOPE + TR1
0369 TFN = DSLN + ESLN * (TN - TSLN)
0370 DUMMY = DN - TFN
0371 IF (DUMMY .GT. 0.0) GOTO 2240
0372 DUMMY = DUMMY*(-1)
0373 2240 IF (NCT .GE. 0.0 .AND. NCT .LE. 1.0 .AND. DUMMY .GT. GTEMP) GOTO 186
0374 GOTO 183
0375 IF (TEN .GE. (TO - GTSW)) GO TO 183
0376 *
0377 C A TEMPERATURE DATUM
0378 *
0379 1821 ICOND(I, JN) = 10 + ICOND(I, JN)
0380 ESLN1 = ESLN
0381 ESLN = 0.8*ESLN + 0.2*(DN - DSLN)/(TN - TSLN)
0382 DUMMY = ESLN - ESLN1
0383 IF (DUMMY .GT. 0.0) GOTO 2250
0384 DUMMY = DUMMY*(-1)
0385 2250 IF (DUMMY .GT. 0.0) ESLN = ESLN1
0386 DSLN = 0.2*DSLN + 0.2*DN
0387 TSLN = TN
0388 INCHN = 1
0389 GO TO 1874
0390 C BACK-ASSIGN CHANNEL AND CONTACT NUMBERS
0391 *
0392 183 DUMMY = TN - TO
0393 DUMMY1 = DN - TFN
0394 IF (DUMMY .GT. 0.0) GOTO 2260
0395 DUMMY = DUMMY*(-1)
0396 2260 IF (DUMMY1 .GT. 0.0) GOTO 2270
0397 DUMMY1 = DUMMY1*(-1)
0398 2270 IF (DUMMY1 .GE. 0.0 .OR. DUMMY1 .LE. GTEMP) GOTO 1873
0399 IF (NCT .GE. 5 .OR. NCT .LE. 0) GO TO 1873
0400 IF (ICOND(2, JN1)) = 1.0 .EQ. ICOND(2, JN1) + NCT) GOTO 1873
0401 ICOND(2, JN) = (I.JUCNR(2, JN)) + NCT)
0402 *
0403 C AN HUMIDITY DATUM
0404 *
0405 ICOND(1, JN) = 40 + ICOND(1, JN)
0406 INCHN = 4
0407  GO TO 1874
0408  
0409  1873 IF(ICOND(1,JN) .EQ. 1) GO TO 1821
0410  186  ICOND(1,JN) = MM + ICOND(1,JN)
0411  #1874 IF(ICOND(2,JN)/1000. .EQ. 0) ICOND(2,JN)=ICOND(2,JN)+MM*100 ;(1874)
0412  1874 CONTINUE
0413  1875 CONTINUE
0414  
0415  C WRAF-UP PROCESS -E-
0416  
0417  188 IF(M1 .LT. 0) GO TO 189
0418  189  ICR1 = ICR
0419  190  SLOPE1 = SLOPE
0420  
0421  189 JKR1 = JKR
0422  189 TR1 = TR
0423  189 T2 = TR
0424  189 SLOPE2 = SLOPE
0425  189 M1 = 1
0426  189 KROSS = 0
0427  189 JINSTRT = JK + 1
0428  189 GO TO 78
0429  C  
0430  C STORE REFERENCE DATA
0431  C  
0432  200 CONTINUE
0433  200 IF(INCH .EQ. 0) GO TO 220
0434  201 IF(INCH .EQ. 2) GO TO 220
0435  C  
0436  C A CONTACT SWITCH POINT
0437  
0438  204 JR = JK
0439  204 RFSUM = 0.0
0440  204 RTSUM = 0.0
0441  C  
0442  C CUMULATION OF REFERENCE MEAN FREQUENCY
0443  C  
0444  220 RFSUM = RFSDUM + D * DWELL
0445  221 RTSUM = RTSUM + DWELL
0446  222 ICOND(1,J) = 2
0447  223 INCH = 2
0448  C  
0449  900 IF(IDG .LE. 2) RETURN ;900
0450  901 IF(MOD(ITCNT,50) .EQ. 0) WRITE(IOUT,9001)
0451  902 ITCNT = ITCNT + 1
0452  903 WRITE(IOUT,9002)(ICOND(I,J),I=1,3), (ICOND(I,J),I=1,2), JK,
0453  904 INCH, NJH, KFL, DFLFL, TF, FHUM, ESL, SLOPE, KROSS, M1
0454  905 FORMAT('1DECOM OUTPUT: ------ICOND(I,J),I=1,3',
0455  905 & '------ICOND JK INCH NOH RFL',
0456  906 & ' ------DJEM DFLFL TF FHUM ESL SLOPE KROSS M1')
0457  900 FORMAT('DECOM OUTPUT: ',3F7.1,S15,4F7.1,F7.4,F7.2,2I6)
0458  908  
0459  901 RETURN
0460  C  
0461  END
This program is used to format the data stored in bytes 1-10 during an interrupt. Normal operation is to do this format during the interrupt, but if we are bringing data into the main program we don't want to start changing the data stack or pointers. To solve the problem a flag is set that tells the interrupt not to format the data. Then after the data is in the FORTRAN program it sees if INTMSK (the flag) has been set to 2 by an interrupt. If so this program is called.

CSCT

IFLAG RMB 2
BYTE10 RMB 1
BYTE9 RMB 1
BYTE8 RMB 1
BYTE7 RMB 1
BYTE6 RMB 1
BYTE5 RMB 1
BYTE4 RMB 1
BYTE3 RMB 1
BYTE2 RMB 1
BYTE1 RMB 1
BYTE11 RMB 1
BYTE12 RMB 1
BYTE13 RMB 1
BYTE14 RMB 1
BYTE15 RMB 1
BYTE16 RMB 1
TEMPX RMB 2
INTMSK RMB 2
IRAM1 RMB 2
IRAM2 RMB 2
IRAM3 RMB 2
IRAM4 RMB 2
IRAM5 RMB 2
IRAM6 RMB 2
IRAM7 RMB 2
IRAM8 RMB 2
IRAM9 RMB 2
IRAM10 RMB 2
ITOP RMB 2
IBOT RMB 2
IDATA RMB 1500

IRAM are stacked into the IDATA array to be processed

PROGRAM follows

RAW DATA IS NOW STORED FROM BYTE1-BYTE10

THIS WAS DONE IN THE INTERRUPT.

FORMAT THE DATA INTO INTEGER IN IRAM USING BYTE11-BYTE16
AS SCRATCH PAD, THEY ARE STORED AS FOLLOWS:

- IRAM1=HOURS
- IRAM2=MIN
- IRAM3=SEC
- IRAM4=.SEC
- IRAM5=DEGREES OF EL
- IRAM6=DEGREES OF EL
- IRAM7=DEGREES OF AZ
- IRAM8=DEGREES OF AZ
- IRAM9=THOUSANDS PLACES IN MET WORD
- IRAM10=LOWEST 3 VALUES OF MET WORD (IL UP TO 999)

START BY FINDING VALUE OF HOURS

LDE BYTE10
LSRB
LSRB
STA BYTE11
LSRB
LSRB
LSRB
LSRB
LDA BYTE11
ANDA #$0F
STA BYTE11
LDA BYTE9
ASLA
BCC NOCAR
LDA BYTE11
LSLA
ORA #$01
STA BYTE12
BRA LOWMIN
NOCAR LDA BYTE11
LSLA
STA BYTE12
LOWMIN LDA BYTE9
ANDA #$7F
LSRA
LSRA
LSRA
STA BYTE13
LDE BYTE12
LDA #$0A
MUL
ADDB BYTE13
STC IRAM2

NOW DO SECONDS
PAGE 003 GETBCK .SA:1 UMET-1  15 DEC 1982

0117  LDA  BYTEE
0118  LSRA
0119  LSRA
0120  LSRA
0121  LSRA
0122  STA  BYTE11
0123  LDE  BYTE9
0124  ANDB  #$07
0125  LDA  #$0A
0126  MUL
0127  ADDB  BYTE11
0128  STD  IRAM3
0129  *    NOW DO TENTHS OF SECONDS
0130  LDE  BYTEE
0131  ANDB  #$0F
0132  LDA  #$00
0133  STD  IRAM4
0134  *
0135  *    TIME WORD DONE
0136  *    START ON EL WORD
0137  LDA  BYTEE
0138  ANDA  #$0F
0139  STA  BYTE11
0140  LDE  BYTE7
0141  ANDB  #$F0
0142  LSRB
0143  LSRB
0144  LSRB
0145  LSRB
0146  LDA  #$0A
0147  MUL
0148  ADDB  BYTE11
0149  STD  IRAM5
0150  *
0151  LDA  BYTE6
0152  ANDA  #$0F
0153  STA  BYTE11
0154  LDE  BYTE6
0155  ANDB  #$F0
0156  LSRB
0157  LSRB
0158  LSRB
0159  LSRB
0160  LDA  #$0A
0161  MUL
0162  ADDB  BYTE11
0163  STD  IRAM6
0164  *
0165  *
0166  LDE  BYTES
0167  LSRB
0168  LSRB
0169  LSRB
0170  LSRB
0171  LDA  #$64
0172  MUL
0173  STD  BYTE13
0174  LDE  BYTES
0175 ANDE #$0F
0176 LDA #$0A
0177 MUL
0178 ADDD BYTE13
0179 STD BYTE13
0180 LDA #$00
0181 LDB BYTE4
0182 LSRA
0183 LSRA
0184 LSRA
0185 LSRA
0186 ADDD BYTE13
0187 STD IRAM7
0188 * NOW DO TENTHS OF DEGREE AZ
0189 LDA BYTE3
0190 LSRA
0191 LSRA
0192 LSRA
0193 LSRA
0194 STA BYTE11
0195 LDB BYTE4
0196 ANDB #$0F
0197 LDA #$0A
0198 MUL
0199 ADDD BYTE11
0200 STD IRAM8
0201 * AZ WORD DONE
0202 * START ON MET WORD
0203 LDA BYTE2
0204 LSRA
0205 LSRA
0206 LSRA
0207 LSRA
0208 STA BYTE11
0209 LDB BYTE3
0210 ANDB #$0F
0211 LDA #$0A
0212 MUL
0213 ADDD BYTE11
0214 STD IRAM9
0215 * DO LAST OF MET WORD
0216 LDB BYTE2
0217 ANDB #$0F
0218 LDA #$64
0219 MUL
0220 STD BYTE13
0221 LDB BYTE1
0222 ANDB #$0F
0223 STB BYTE11
0224 LDB BYTE1
0225 LSRA
0226 LSRA
0227 LSRA
0228 LSRA
0229 LDA #$0A
0230 MUL
0231 ADDD BYTE11
0232 ADDD BYTE13
DATA IS FORMATTED AS INTEGERS
NOW FORMATT INTO IDATA ARRY FOR USE IN MAIN
MET PROGRAM.

LDD ITOP
ADDD #$01
CMPD #$150
BNE ITOP
LDD #$0000
ISTOP STD ITOP
CMPD IBOT
BNE SETFLG
LDD IBOT
ADDD #$01
CMPD #$150
BED ATTOP
STD IBOT
BRA SETFLG
ATTOP LOD #S0000
STD IBOT
SETFLG LDD ITOP
CMPD IBOT
BLD WORKUP
SUBD IBOT
STD IFLAG
BRA STOR
WORKUP ADDD #$150
SUBD IBOT
STD IFLAG
STOR LDX TEMPX
LDD IRAM1
STD ,X++
LDD IRAM2
STD ,X++
LDD IRAM3
STD ,X++
LDD IRAM4
STD ,X++
LDD IRAM5
STD ,X++
LDD IRAM6
STD ,X++
LDD IRAM7
STD ,X++
LDD IRAM8
STD ,X++
LDD IRAM9
STD ,X++
LDD IRAM10
STD ,X++
CMPX #BBCE6 ;HIGHEST ADDRESS OF DATA STACK
SHS LOOPFX
STX TEMPX
BRA DONE
LOOPFX LDX #BB12E ;BOTTOM ADDRESS OF DATA STACK
STX TEMPX
PAGE 006  GETBECK  .SA:;1  UMET-1  15 DEC 1982
0271  LDD  #100
0272  STD  ITOP
0273  DOWE  PULS  X
0274  RTS
0275  END

;RESTORES FORTRAN X REGISTER
SUBROUTINE INTERF(JK, TNOH, ISTOP, LCNT)

COMMON (LOCK :

INCLUDE 'COMMON.SA'

DIMENSION T1(7), T2(7), VI(7), ALOSS(7)

TOLERABLE TIME INTERVALS BETWEEN SIGNAL DATA
(P, R, T, H)

ALoss(1)=0.0
ALoss(2)=0.0
ALoss(3)=0.0
ALoss(4)=200.0
ALoss(5)=600.0
ALoss(6)=100.0
ALoss(7)=100.0
KNTCT = 0
I4 = 1
I5 = 1
I6 = 1
I7 = 1
DO 1 I = 4,7
1 VI(I) = 0,
1 TLFCAL = 1.0E10

DO EACH ROW (TIME) OF OUTPUT TABLE

DO 30 L = 1, LIST
30 IF(VL(1,L) .GE. TBRST) GO TO 47
IF(VL(1,L) .GE. TLFCAL) GO TO 42
IF(VL(1,L) .GE. COND(1,JK)) GO TO 48

DO EACH COLUMN ENTRY (VARIABLE) OF THE OUTPUT TABLE

IF TL IS BRACKETED, INTERPOLATE

DO 20 IV = 4,7
20 IF(VL(1,L) .LE. T2(IV)) GO TO 101

ADVANCE BRACKET BEFORE INTERPOLATING

I5 = IV - 3
GO TO (11, 12, 13, 14), I5

NEXT PRESSURE PAIRS

11 DO 111 I = I4, 900
IF(I .GT. JK) GO TO 101
IF(ICOND(2, I) .LT. ICOND(1, I)) GO TO 112
CONTINUE
CONTINUE

CONTINUE

NEXT REFERENCE FREQUENCY PAIRS

NEXT TEMPERATURE PAIRS

NEXT RELATIVE HUMIDITY PAIR

PAGE 002 INTERP .SA:11 UMET-1 15 DEC 1982

0059 112 CONTINUE
0060 KNTCT = ICOND(2,I)
0061 T1(4) = T2(4)
0062 V1(4) = V2(4)
0063 V2(4) = FCAL(KNTCT)
0064 T2(4) = COND(1,I)
0065 I4 = I + 1
0066 GO TO 10
0067 *
0068 *
0069 * NEXT REFERENCE FREQUENCY PAIRS
0070 12 DO 121 I = 15,900
0071 IF(I .GT. JK .OR. COND(1,I) .GT. THERST) GO TO 101
0072 IF(ICOND(2,I) .EQ. 0) GO TO 121
0073 IF( ICOND(1,I) .EQ. 2) GO TO 122
0074 121 CONTINUE
0075 122 CONTINUE
0076 T1(5) = T2(5)
0077 V1(5) = V2(5)
0078 V2(5) = COND(3,I)
0079 T2(5) = COND(1,I)
0080 I5 = I + 1
0081 GO TO 10
0082 *
0083 *
0084 * NEXT TEMPERATURE PAIRS
0085 13 DO 131 I = 16,900
0086 IF(I .GT. JK .OR. COND(1,I) .GT. THERST) GO TO 101
0087 IF(ICOND(1,I) .EQ. 1 .AND. COND(3,I) .GT. 0.001) GO TO 132
0088 131 CONTINUE
0089 132 T1(6) = T2(6)
0090 V1(6) = V2(6)
0091 V2(6) = 95. * COND(3,I)/ VL(5,L)
0092 T2(6) = COND(1,I)
0093 I6 = I + 1
0094 GO TO 10
0095 *
0096 * NEXT TEMPERATURE PAIRS
0097 133 CONTINUE
0098 V1(6) = VL(6,L-2)
0099 T1(6) = VL(1,L-2)
0100 V2(6) = VL(6,L-1)
0101 T2(6) = VL(1,L-1)
0102 GO TO 10
0103 *
0104 * NEXT RELATIVE HUMIDITY PAIR
0105 14 IF(VL(1,L) .GT. TNOH) GO TO 20
0106 DO 141 I = 17,900
0107 &
0108 IF(I .GT. JK .OR. COND(1,I) .GT. TNOH .OR. COND(1,I) .GT. THERST) GO TO 101
0109 *
0108 IF( ICOND(1,I) .EQ. 4 .AND. COND(3,I) .GT. 0.001) GO TO 142
0109 * CONTINUE
0110 142 T1(7) = T2(7)
0111 V1(7) = V2(7)
0112 V2(7) = 95. * COND(3,I)/ VL(5,L)
0113 T2(7) = COND(1,I)
0114 I7 = I + 1
0115 GO TO 10
0116 *
INTERPOLATE / EXTRAPOLATE

101 IF(IV .EQ. 7 .AND. VL(1,L) .GT. TNOH) GO TO 20
102 IF( V1(IV).GT. .0001 .AND. (T2(IV) - T1(IV)) .GT. .0001) GO TO 190
103 VL(IV,L)= V2(IV)
104 GO TO 20
105 VL(IV,L)=V1(IV)*((V2(IV)-V1(IV))/(T2(IV)-T1(IV)))*(VL(1,L)-T1(IV))
106 C
107 DUMMY = VL(1,L)-T1(IV)
108 DUMMY1 = VL(1,L)-T2(IV)
109 IF (DUMMY .GT. 0) GOTO 2210
110 DUMMY = DUMMY*(-1)
111 IF (DUMMY1 .GT. 0) GOTO 2200
112 DUMMY1 = DUMMY1*(-1)
113 IF(DUMMY .GT. ALOSS(IV)) AND,
114 & DUMMY1 .GT. ALOSS(IV) ) VL(IV,L) = 0.
THIS PROGRAM ALLOWS REAL TIME INTERRUPTS TO OCCUR

* PROGRAM FOLLOWS

**E002

TO SAVE THE FORTRAN X REG AND USE IT AT E002 IN THIS PROG
0059  STA 16,X ;PIA CONTROL REGISTER
0060  STA 17,X ;PIA CONTROL REGISTER
0061  LDA 1,X
0062  ORA #07 ;TD CONFIGURE FOR INTERRUPTS THROUGH PIA
0063  STA 1,X
0064  LDD #$0000 ;SO IFLAG WILL BE RESET
0065  STD IFLAG
0066  LDD #$5620 ;ADDRESS OF INTERRUPT PROGRAM
0067  STD >$FFFE ;PSEUDO IRQ INTERRUPT VECTOR
0068  LDD BYTE17 ;X REG STORED AT BYTE17
0069  TFR D,X ;RESTORES THE X REG FOR FORTRAN
0070  LDD #$B12E
0071  STD BYTE17 ;SET UP POINTER FOR IDATA STACK
0072  LDA #$E007
0073  ORA #$38
0074  STA #$E007 ;TURNS ON THE SHIFT REGISTERS
0075  ANDCC #$EF ;CLEASES INTERRUPT ILAG
0076  RTS
0077  END
Instructions to users:

1. Change the I/O address definitions below to the correct values for your system.
2. Assemble this file as follows:
   - RASM IDPKG: -FRXLN=80
3. In "LOAD", load this module (IDPKG.RO) before doing a FORTRAN library (FORLB.RO) search.

If your monitor already has the following EXBUG equivalent subroutines, then memory size may be conserved by putting the following routine into a separate file (IOADRS) and changing the entry locations. Assemble this file as follows:
   - RASM IOADRS: -RXLN=80

Load this module (IOADRS.RO) before doing a FORTRAN library search. Or you may specify the addresses at link time via the "DEF" command before doing a FORTRAN library search.

Example:

-LOAD
 ?DEF:INSNP=$F015
 ?DEF:OUTCH=$F018
 ?DEF:PCRLF=$F021
 ?DEF:PDATI=$F027
 ?DEF:LOUTC=$EBCC
 ?BASE
 etc.

PAGE

EQUATES

CR EQU $0A
LF EQU $0D
I

PACE 002 IOPKG SA:1 UMET-1 15 DEC 1982

0059 EOT EQU $04
0060 SKIP2 EQU $8C SKIP 2 LOCATIONS
0061 *(CFX * DP CODE)
0062 SPC 1
0063 I/O ADDR DEFS I/O ADDR
0064 ACIAI EQU ECI4 Resident monitor's ACIA addr
0065 ACIAI EQU ACIAI+ Input ACIA base address
0066 $FCF4= EXBUG compatible
0067 $= 0= Status, +1= Data
0068 ,CTRLI EQU %00100011 Input ACIA ctrl reg byte
0069 # Divide by 16 clock
0070 # 8-bits + 2 stop bits
0071 # RTS low, interrupts disabled
0072 SPC 1
0073 ACIAO EQU ACIAO+ Output ACIA base address
0074 $FCF4= EXBUG compatible
0075 +0= Status, +1= Data
0076 ,CTRLO EQU %00100011 Output ACIA ctrl reg byte
0077 # Divide by 16 clock
0078 # 8-bits + 2 stop bits
0079 # RTS low, interrupts disabled
0080 SPC 1
0081 LPIA EQU $EC10 Lineprinter PIA base address
0082 $EC10= EXBUG compatible
0083 +0= Data (A) +1= Control 1
0084 +2= Status (B) +3= Control 2
0085 # Bit 0=Select
0086 # Bit 1=No paper
0087 ,CTRLA EQU %01111100 LP "A" ctrl reg byte
0088 ,CTRLB EQU %01111100 LP "B" ctrl reg byte
0089 ,STRAF EQU %01101100 LP "A" ctrl reg strobe
0090 $3C,E3C,S34= EXBUG/Centronics compatible
0091 * Note: If a serial line printer is to be used, make
0092 * the following changes:
0093 1) Change ,LPIA to ,LACIA and substitute
0094 2) Change ,CTRLA to ,ACIAL and substitute
0095 proper control reg byte.
0096 3) Delete ,CTRLA and ,STRAF lines.
0097 4) Replace the LPOUTC subroutine with a
0098 matching serial driver.
0099 5) Change INITLZ subroutine starting at
0100 INIT3 to initialize the line printer
0101 * ACIA.
0102 PAGE
0103 * Monitor I/O address table for FORTRAN I/O
0104 * named common PSCT so user can easily overlay
0105 * with other I/O addresses by loading last.
0106 SPC 1
0107 * IOADR COMM PSCT
0108 ACIAI FDB .ACIAI Input ACIA address
0110 CTRLI FCB .CTRLI Input ACIA ctrl reg byte
0111 ACIAO FDB .ACIAO Output ACIA address
0112 CTRLA FCB .CTRLA Output ACIA ctrl reg byte
0113 LPIA FDB .LPIA Lineprinter PIA address
0114 CTRLA FCB .CTRLA LP FIA ctrl reg byte
0115 CTRLB FCB .CTRLB LP FIA ctrl reg byte
0116 STRA FCB .STRB LP FIA ctrl reg byte

DEC 1982
Initialize FORTRAN I/O Subroutine

Entry:

X = base ACIA address
E = control register byte

Initialize ACIA Subroutine

%171 .INITA LDA 3
%172 STAA 0, X Master reset
%173 STAB 0, X Set control register
%174 RTS
PAGE 009 IOPKG .SA11 UMET-1 15 DEC 1982

0175 SPC 1
0176 ****************************************
0177 x Print CR,LF,NULL Subroutine
0178 x uses A register
0179 x preserves E and X
0180 ****************************************
0181 FCRLF$ PSHB
0182 LDAA @CR
0183 BSR OUTCH$
0184 LDAA @LF
0185 BSR OUTCH$
0186 PULS E,PC RETURN
0187 SPC 1
0188 ****************************************
0189 x Input char, strip parity, & echo subroutine
0190 x Exit: A= char (7-bits)
0191 x E and X preserved
0192 ****************************************
0193 INMP JSR INMP$
0194 x Fall into char output subroutine
0195 SPC 1
0196 ****************************************
0197 x Character Output Subroutine
0198 x A= Char to output (8-bits)
0199 x A, E and X preserved
0200 ****************************************
0201 OUTCH$ PSHB A,E,X Save reg's
0202 OUTCH1 LDX ACIA0$
0203 LDAE 0,X Get ACIA status
0204 ANDE #2 Ready yet?
0205 BEO OUTCH4 No, try again
0206 LDA 0,S
0207 STA 1,X Yes, send char
0208 LDAE CHNUL Char nulls
0209 x Test for CR nulls or char nulls
0210 CMPA #CR
0211 ENE OUTCHS
0212 LDAB CRNUL CR nulls
0213 OUTCH5 TSTS
0214 EEO OUTCH7 No, try again
0215 CLRA
0216 STA 1,X Send null
0217 BRA OUTCH$
0218 PAGE
0219 ****************************************
0220 BRA OUTCH$
0221 SPC 1
0222 OUTCH1 PULS E,PC Restore reg's & return
0223 SPC 1
0224 OUTCH4 LDX GTMAIN
0225 BEO OUTCH$
0226 TST CID$ MM12 Controller in charge?
0227 JSR 0,X Do possible handshake
0228 BRA OUTCH$
0229 PULS A,B,\',PC Restore reg's & return
0230 SPC 1
0231 OUTCH4
0233 * Print Message String Subr. (no CR, LF)
0234 * X=adr of data string
0235 * uses A and X reg's
0236 * preserves E
0237 ******************************************
0238 PDAT1* LDAA 0,X Get next char
0239 CMFA #EDT
0240 BEG RTKN
0241 BSR OUTCH* Output char
0242 INX Point to next one
0243 BRA PDAT1* Continue
0244 SPC 1
0245 ******************************************
0246 * Input Char (w/parity, no echo) Subroutine
0247 * Exit: A= char (8-bits) E and X preserved
0248 *
0249 ******************************************
0250 IMNE PSHS X
0251 INCH0 LDX ACIA1*
0252 LDAA 0,X Get ACIA status
0253 ASRA Ready?
0254 BCC INCH2
0255 LDAA 1,X Yes, got char
0256 PULS X,PC Return
0257 SPC 1
0258 INCH2 LDX GTMAIN
0259 BEG INCH0
0260 TST CIC*
0261 BNE INCH0
0262 PSHR
0263 JSR 0,X
0264 PULB
0265 BRA INCH0
0266 PAGE
0267 ******************************************
0268 * Output Char to LF Subroutine
0269 * Entry: A= char to print (8-bits)
0270 * Exit: C= 1 if error
0271 *
0272 ******************************************
0273 LOUTC* PSHS X
0274 LDX GTMAIN
0275 BEG LOUT0
0276 TST CIC*
0277 BEG LOUT10
0278 LOUT0 LDX LFIA1*
0279 STAA 0,X Send data
0280 LDAA 0,X Clear acknowledge
0281 LDDA STREA1*
0282 BSR STROBE Send strobe
0283 LOUT1 LDAA 2,X Check status
0284 ANGA #3 Bit 0=Select,
0285 *
0286 DECA A Should have been #01
0287 BNE LOUTER No paper or not ready
0288 TST 1,X Acknowledge?
0289 BPL LOUT1 NO
0290 CLC
PAGE 006 1D0KG .SA11 1UMET-1 15 DEC 1962

0291 LOUTX LDAA 0,X
0292 FULS X,FC Return
0293 SPC 1
0294 LOUTER SEC
0295 BRA LOUTX
0296 SPC 1
0297 LOUT10 FSHS A,B
0298 JSR 0,X
0299 FULS A,B
0300 BRA LOUT0
0301 SPC 1
0302 ****************************************
0303 * Strobe Printer Subroutine *
0304 * A= Strobe value
0305 * B and X preserved
0306 ****************************************
0307 STROBE BSR STROB1 Send strobe byte
0308 LDAA CTRLA$ Reset strobe
0309 STROB1 STAA 1,X
0310 RTS
0311 SPC 1
0312 ****************************************
0313 * Input Char (strip parity, no echo) Subr.
0314 * Exit: A= char (7-bits)
0315 * E and X preserved
0316 ****************************************
0317 IN$NF'E BSR IN$NE
0318 ANDA #$7F Strip parity
0319 RTS
0320 PAGE
0321 END
0001 OPT L,REL
0002 OPT LLE=00
0003 NAM JUMPER
0004 IDNT JUMPER
0005 XDEF JUMPER
0006 *
0007 THIS PROGRAM RETURNS TO THE MONITOR
0008 *
0009 CSCT
0010 IFLAG RMB 2
0011 BYTE10 RMB 1
0012 BYTE9 RMB 1
0013 BYTE8 RMB 1
0014 BYTE7 RMB 1
0015 BYTE6 RMB 1
0016 BYTE5 RMB 1
0017 BYTE4 RMB 1
0018 BYTE3 RMB 1
0019 BYTE2 RMB 1
0020 BYTE1 RMB 1
0021 BYTE11 RMB 1
0022 BYTE12 RMB 1
0023 BYTE13 RMB 1
0024 BYTE14 RMB 1
0025 ISTPER RMB 2
0026 TEMPX RMB 2
0027 INTMSK RMB 2
0028 IRAM1 RMB 2
0029 IRAM2 RMB 2
0030 IRAM3 RMB 2
0031 IRAM4 RMB 2
0032 IRAM5 RMB 2
0033 IRAM6 RMB 2
0034 IRAM7 RMB 2
0035 IRAM8 RMB 2
0036 IRAM9 RMB 2
0037 IRAM10 RMB 2
0038 ITOF RMB 2
0039 IEOF RMB 2
0040 IDATA RMB 1500 ;IRAM ARE STACKED INTO THE IDATA ARRY TO BE PROCESSED
0041 ****************************************
0042 *
0043 *
0044 PROGRAM FOLLOWS
0045 *
0046 PSCT
0047 JUMPER LDA >$E003
0048 ANDA $FC
0049 STA >$E003 ;TURNS OFF INTERRUPTS
0050 JMP >$FB30 ;JUMPS TO MONITOR
0051 END
DECOMMUTATOR PROGRAM - UNIV. OF UTAH

OPTION STACK=200

**COMMON BLOCK**

DIMENSION LINE(40),LDATE(3),LTIME(3),ID(3),ISTT1(3),ISTT2(3)

`SEARCH AND GATE HWIDTHS`

WRITE(100,5431)
5431 FORMAT(//, 'UNIVERSITY OF UTAH', //)
5432 FORMAT('REAL TIME PROCESSING OF METEOROLOGICAL BALLOON SOUNDING DATA', //)
5433 FORMAT(13)
5434 WRITE(101,5432) I
5435 FORMAT(I3)
DO 13 J=1,120
DO 19 I=1,7
VL(I,J) = 0.0
CONTINUE
13 CONTINUE
19 CONTINUE
CALL PRINTER ;SET UP PRINTER FOR 132 CHAR LINE
IOUT=102
WRITE(IOUT,9998)
9998 FORMAT('UNIV. OF UTAH AUG 1982',//)
WRITE(IOUT,9999)
9999 FORMAT(1X,45X,'INPUT DATA **********'/)
IOUT=100
WRITE(IOUT,10004)I1,I2,TS3

**DECOMMUTATOR PROGRAM**

TPROC=9999.0
TSTART=0.
GOTO 5108
0059 10094 FORMAT(' BALLOON RELEASED AT ',2(I2,'.'),F4.1)
0060 C
0061 C CONVERT II(HOURS),I2(MIN),TS3(SEC) TO SECONDS
0062 C
0063 TLANCH = I1*3600. + I2*60. + TS3
0064 IF (TSTART .LT. .01) TSTART = - 120.
0065 IF (TPROC .LT. .01) TPROC = 10000.
0066 TSTOP = TPROC - TSTART
0067 GOTO 6764
0068 C
0069 C *** LINENO IS THE NUMBER OF LINES PER PAGE TO BE PRINTED
0070 C
0071 5108 LINENO=54
0072 C
0073 C INITIALIZE CONDENSER
0074 C
0075 C FSUM = 0.0
0076 NSUM = 0
0077 C LOSS OF SIGNAL FLAG LOS
0078 C
0079 LOS = 0
0080 C MODE INTERVAL ( OVERLAPPING BANDS) HALF-WIDTH ( 0.5 Hz)
0081 C
0082 HGATE = 1.0
0083 C SIGNAL RANGE ( SIGMIN TO SIGMAX Hz)
0084 C
0085 SIGMIN = 5.
0086 SIGMAX = 205.
0087 IN = (( SIGMAX - SIGMIN )/ HGATE ) + 1
0088 C CONDENSED DATA INDEX JK, FOR ONE DECOMMUTATION CYCLE
0089 C
0090 JJ=0
0091 J=0
0092 ISTPER=0
0093 C
0094 DO 1 I=1,900
0095 COND(1,I) = 0.0
0096 COND(2,I) = 0.0
0097 COND(3,I) = 0.0
0098 ICOND(1,I) = 0
0099 ICOND(2,I) = 0
0100 CONTINUE
0101 C
0102 C INITIAL EXPECTED SIGNAL LEVELS
0103 C
0104 JJ=0
0105 DO 1 I=1,900
0106 VL(I,JJ) = 0.0
0107 CONTINUE
0108 C
0109 C INITIALIZE TABLE
0110 C
0111 TNOH = 10000.0
0112 VL(1,1) = 0.0
0117  DLIST=60.
0118  TGMDAO=20.
0119  LIST=1
0120  WRITE(100,9811)
0121  9811  FORMAT('ENTER STATION ID (XX)',/)  
0122  READ(101,5432)IDSTAT
0123  WRITE(100,9816)
0124  9816  FORMAT('ENTER STATION GEOPOTENTIAL HEIGHT (XXXX.X)',/)  
0125  READ(101,5444)HEIGHT
0126  WRITE(100,9806)
0127  9806  FORMAT('ENTER ZERO AZIMUTH: NORTH = 0, SOUTH = 1 ',/)  
0128  READ(101,5432)IAZU
0129  WRITE(100,9802)
0130  9802  FORMAT('ENTER LAUNCH MONTH (XX)',/)  
0131  READ(101,5433)IMONTH
0132  WRITE(100,9814)
0133  9814  FORMAT('ENTER LAUNCH DAY (XX)',/)  
0134  READ(101,5433)IDAY
0135  WRITE(100,9803)
0136  9803  FORMAT('ENTER LAUNCH YEAR (XX)',/)  
0137  READ(101,5433)IYEAR
0138  WRITE(100,9804)
0139  9804  FORMAT('ENTER SURFACE TEMPERATURE IN C (XX.X)',/)  
0140  READ(101,5445)SUFT
0141  WRITE(100,9805)
0142  9805  FORMAT('ENTER SURFACE RELATIVE HUMIDITY % (XX.X)',/)  
0143  READ(101,5495)SURH
0144  WRITE(100,9801)
0145  9801  FORMAT('ENTER SURFACE PRESSURE IN MEAR (XXXX.X)',/)  
0146  READ(101,5494)FP0
0147  WRITE(100,9821)
0148  9821  FORMAT('ENTER SURFACE WIND SPEED (XXX.X)',/)  
0149  READ(101,5467)VSFC
0150  WRITE(100,9823)
0151  9823  FORMAT('ENTER SURFACE WIND DIRECTION (XXX.X)',/)  
0152  READ(101,5467)DSFC
0153  WRITE(100,9812)
0154  9812  FORMAT('ENTER SONDE ID NUMBER - LESS THAN 32000 (XXXXX)',/)  
0155  READ(101,5459)IDSOND
0156  WRITE(100,9813)
0157  9813  FORMAT('ENTER UNADJUSTED REFERENCE ORDINATES (XX.X)',/)  
0158  READ(101,5445)FR
0160  WRITE(100,9817)
0161  9817  FORMAT('ENTER AIR TEMP CALIBRATION ORDINATES (XXX.X)',/)  
0162  READ(101,5467)RECTF
0163  WRITE(100,9818)
0164  9818  FORMAT('ENTER REL HUM CALIBRATION ORDINATES (XXX.X)',/)  
0165  READ(101,5467)CALRH
0166  WRITE(100,6136)
0167  6136  FORMAT('ENTER SURFACE TEMP ORDINATES (XXX.X)',/)  
0168  READ(101,5467)FTEMP
0169  WRITE(100,6137)
0170  6137  FORMAT('ENTER SURFACE RH ORDINATES (XXX.X)',/)  
0171  READ(101,5467)FRH
0172  WRITE(F6.1)
0173  F6.1  FORMAT(F6.1)
0174  5467  FORMAT(F5.1)
0175 WRITE(100,9819)
0176 9819 FORMAT(175,'FOR MILITARY SONDE (ML419,ML418) ENTER 1',/,
0177 & 'FOR STANDARD NWS SONDE ENTER 2',/)
0178 READ(101,5433)ICBN
0179 IF (FCBN .LT. .01) GO TO 6665
0180 RFL=2.0*FR59-10.0
0181 PFL=RFL+50
0182 GOTO 6666
0183 6665 FR=95.0
0184 RFL=170.
0185 6667 PFL=190.
0186 6666 V2( 4) = FP0
0187 V2( 5) = FR0+2.
0188 V2( 6) = FTEMPO
0189 V2( 7) = FRHD
0190 C
0191 C
0192 C
0193 C
0194 CNVDF = 2.*FR0/95.
0195 IF (CNVDF .LT. .01) GO TO 5
0196 TF = FTEMPO*CNVDF
0197 5 CONTINUE
0199 X
0200 X
0201 X
0202 TBRST = 1.E22
0203 WRITE(100,9805)
0204 9805 FORMAT('ENTER CONTACT NUMBER AT BURST (XXX.X)',/,
0205 & 'IF < 10 IS INPUT, VALUE DEFAULTS TO 180.01',/)
0206 READ(101,5446)CBRS1
0207 5446 FORMAT(F5.1)
0208 IF (CBRS1 .LT. 10.) CBRST = 180.01
0209 HEIGHT=HEIGHT+.0009 ;TO ACCOUNT FOR TURNINATION
0210 FP0=FP0+.0009
0211 FTEMP0=FTEMP0+.0009
0212 FR=FR0+.0009
0213 RECTF=RECTF+.0009
0214 CALRH=CALRH+.0009
0215 WRITE(IOUT,1028)IDSTAT,HEIGHT,IAZU,IMONTH,IDAY,IYEAR,SURT,SURH,
0216 & FP0,VSCF,DSFC,IDSND,FR0,RECTF,CALRH,FTEMP0,FRHD,ICREN.
0217 & CBRST
0218 HEIGHT=HEIGHT-.0009 ;RESTORE INPUT VALUES
0219 FP0=FP0-.0009
0220 FTEMP0=FTEMP0-.0009
0221 FR=FR0-.0009
0222 RECTF=RECTF-.0009
0223 CALRH=CALRH-.0009
0224 1028 FORMAT(175,'STATION ID =',IS,/, 'STATION HEIGHT =',F6.1,/, 
0225 & 'ZERO AZIMUTH =',IS,/, 
0226 & 'DATE =',IS,/, 'SURFACE TEMP =',F6.1,/, 
0227 & 'SURFACE RH =',F6.1,/, 
0228 & 'SURFACE PRESS =',F6.1,/, 
0229 & 'SURFACE WIND SPEED =',F6.1,/, 
0230 & 'SURFACE WIND DIRECTION =',F6.1,/, 
0231 & 'REFERENCE ORD =',F6.1,/, 
0232 & 'RH CAL ORD =',F6.1,/,
I

ISTOP = 0

PRESSURE CALIBRATION INPUT

WRITE(100,8201)
FORMAT(’ENTER PCAL INPUT CODE’,/,’1=USING TAPE READIF’,/,
’2=ENTERING BY HAND’,/)
READ(101,5433)IDUMM
IF(IDUMM .EQ. 1)GOTO 1625
IFLAG=0
DO 12 J=1,180
PCAL(J)=0.0
PCAL(J)=IRAM(1)+.1*IRAM(2)
IFLAG=0
IF(PCAL(J) .LT. .1)GOTO 9881
IF(J .GE. 179)GOTO 9881
GOTO 1627
J=J+1
PCAL(J)-IRAM(1)+(.1*IRAM(2))
IFLAG=0
IF(PCAL(J) .LT. .1)GOTO 9881
IF(J .GE. 179)GOTO 9881.
GOTO 1627
CONTINUE
CALL SETUP
WRITE(140,1626)
FORMAT(’ENTER THE PUNCH PAPER TAPE’)
ISTPER=0
J=0
IF(ISTPER .LT. 1)GOTO 1626
CALL SNPER
IF(ISTPER .LT. 1)GOTO 1625
GOTO 1627
=J+1
PCAL(J)-IRAM(1)+(.1*IRAM(2))
IFLAG=0
IF(PCAL(J) .LT. .1)GOTO 9881
IF(J .GE. 179)GOTO 9881.
GOTO 1627
CALL OFF
GOTO 9882
CONTINUE
DO 980 I=1,180
WRITE(100,9809)I
FORMAT(’ENTER PCAL’.F10.1.)
READ(101,5444)PCAL(I)
CONTINUE
IF(PCAL(I) .LT. .01)GOTO 1623
DIFF1=PCAL(1)-PCAL(2)
PERC=.1+IST/180.
DIFF2=PCAL(I)-PCAL(I-1)
IF(PCAL(1) .LT. 0.0)GOTO 2
IF(PCAL(I+1) .LT. 0.01)GOTO 9
DIFFAV=(DIFF1+DIFF2)/2.0
DIFFHI=DIFFAVx(1. + PERC)
DIFFLD=DIFFAVx(1. - PERC)
IF(DIFF2 .GT. DIFFHI_ .OR. DIFF2 .LT. DIFFLO)GOTO 6
DIFFI=DIFF2
IF(PCAL(I)-DIFFI_ .LT. 0)GOTO 8
IDUMMY-PCAL(I)-DIFF1
IDUMMY = I + 1
WRITE(OUT,100)IDUMMY,PCAL(I+1),DUMMY
FORMAT(’%.’ PCAL(’,.13)’ WAS’.F10.1,’ AND IS NOW’.F10.1)
0291 PCAL(I+1)=PCAL(I)-DIFF1
0292 CONTINUE
0293 WRITE(IOUT,10024)
0294 FORMAT('1','BAROSWITCH PRESSURE CALIBRATION TABLE')
0295 10024 FORMAT(SF10.1)
0296 DO 1631 I=1,180
0297 PCAL(I)=PCAL(I)+.05 ;TO STOP TRUNCATION ERROR
0298 1631 CONTINUE
0299 DO 30 IY = 8,176,8
0300 IDUMMY=IY-7
0301 IDUM=IY-8
0302 WRITE(IOUT,10023) IDUMMY,PCAL(IDUM+1),PCAL(IDUM+2),PCAL(IDUM+3),
0303 & PCAL(IDUM+4),PCAL(IDUM+5),PCAL(IDUM+6),PCAL(IDUM+7),PCAL(IDUM+8)
0304 10023 FORMAT(I3,F10.1)
0305 30 CONTINUE
0306 WRITE(IOUT,10026) (PCAL(IX),IX=177,180)
0307 10026 FORMAT('177:	 ',4F10.1)
0308 DO 1675 I=1,180
0309 PCAL(I)=PCAL(I)-.001; STORE VALUES OF PCAL TABLE
0310 CONTINUE
0311 DO 15 JF = 1,180
0312 ICKO = ((ICRO - PCAL(JF-1))/(PCAL(JF)-PCAL(JF-1)))x100.
0313 ICRO = ICRO + (JF-1)x100
0314 DO 3 LCNTK = 1,180
0315 IF(PCAL(LCNTK) .LT. 0.1) GO TO 4
0316 LCNTK = LCNTK - 1 ;CHANGES TO REAL
0317 AIICRO = DUMMY/100.
0318 WRITE(IOUT,10016)AIICRO,LCNTK
0319 10016 FORMAT(' EFFECTIVE CONTACT NUMBER AT LAUNCH = ','F6.2,/
0320 & HIGHEST CONTACT NUMBER CALIBRATED = ','I3)
0321 WRITE(100,6789)
0322 6789 FORMAT('IF PCAL TABLE IS OK ENTER 1,,IF NOT OK ENTER 2,,/
0323 READ(101,5433)IDUM
0324 5433 IF(IDUM .EQ. 2)GOTO 1623
0325 WRITE(100,6788)
0326 6788 FORMAT('ENTER HOUR OF LAUNCH (XX),/
0327 READ(101,5433)I1
0328 WRITE(100,6787)
0329 6787 FORMAT('ENTER MIN OF LAUNCH (XX),/
0330 READ(101,5433)I2
0331 WRITE(100,6790)
0332 6790 FORMAT('ENTER SECOND OF LAUNCH (XX.X),/
0333 READ(101,5445)TS3
0334 GOTO 1003
0335 WRITE(100,6791)
0336 6791 FORMAT('IF LAUNCH TIME IS OK ENTER 1,,IF NOT OK ENTER 2,,/
0337 READ(101,5433)IDUM
0338 IF(IDUM .EQ. 2)GOTO 6786
0339 GOTO 1003
0340 WRITE(100,6792)
0341 6792 FORMAT('IF LAUNCH TIME IS OK ENTER 1,,IF NOT OK ENTER 2,,/
0342 READ(101,5433)IDUM
0343 IF(IDUM .EQ. 2)GOTO 6786
0344 C FIND TSTART IN RAW DATA
0345 IF(IDG .EQ. 1)IOUT=101
0346 IFLAG=0
0349  ITOP=0
0350  IBOT=0
0351  CALL INTERR
0352  WRITE(100,879)
0353  879 FORMAT('*** SET UP TO RECEIVE REAL TIME DATA ***')
0354  INTMSK=0
0355  ITOP=0
0356  IBOT=0
0357  IFLAG=0
0358  CONTINUE
0359  CALL INTERR
0360  HANDLE REAL TIME DATA
0361  IF (IFLAG .GT. 0) GOTO 9394
0362  CONTINUE
0363  DO 41 J=6,10
0364  4343 IF (IFLAG .GT. 0) GOTO 9394
0365  4344 GOTO 4344
0366  CONTINUE
0367  CALL ADVANC(JJ,TSTOP,TLANCH,TGMDAG,MASK)
0368  IF (MASK .EQ. 1) GOTO 81
0369  IF (MASK .EQ. 2) GOTO 82
0370  IF (MASK .EQ. 3) GOTO 825
0371  IF (MASK .EQ. 4) GOTO 826
0372  JKMEM = JJ+5
0373  WRITE(100,2003) TIME(30)
0374  2003 FORMAT(F7.2,' SECONDS FROM LAUNCH')
0375  IF (TIME(10) .LT. 0.) GOTO 40
0376  CALL ADVANC(JJ,TSTOP,TLANCH,TGMDAG,MASK)
I.OF POOR REL:4U i`

PAGE 008 MAIN .SA:1 UMET-1 15 DEC 1982

0407 CALL TRACK(LOG,JK,MASK)
0408 IF (MASK .EQ. 1)GOTO 83
0409 IF (MASK .EQ. 2)GOTO 85
0410 IF (JKMEM .EQ. JK)GO TO 99
0411 C
0412 CALL DECOM(JK,TM,H,F,ICR0,RFL,F1,F2,1D)
0413 C
0414 IF (COND(1,JK) .GT. TBRST) GO TO 84
0415 GO TO 99
0416 2204 FORMAT (" ERROR IN CONOPASS#. READ")
0417 C
0419 C
0420 825 WRITE(IOUT,1804) ISTOF=11
0421 GOTO 90
0422 826 WRITE(IOUT,1806) ISTOF=12
0424 GOTO 90
0425 81 WRITE(IOUT,1810) TSTOP,TIME(IO),JJ
0427 ISTOF = 7
0428 GO TO 90
0429 82 WRITE(IOUT,1820) TIME(IO), JJ
0430 ISTOF = 6
0431 GO TO 90
0432 63 WRITE(IOUT,1830) LOS
0433 ISTOF = 5
0434 GO TO 90
0435 85 WRITE(IOUT,1850)
0436 ISTOF = 8
0437 GO TO 90
0438 84 WRITE(IOUT,1840) JK,COND(1,JK),TBRST
0439 1810 FORMAT(2X, 'TSTOP,TIME(IO),JJ =', 2F10.1, 1110)
0440 1820 FORMAT(2X, 'END OF FILE,TIME(IO),JJ =', 10X, F10.1, 1110)
0441 1830 FORMAT(2X, ' LOS =', 10X, I6)
0442 1840 FORMAT(' TIME EXCEEDS TBRST...COND(1,'',13,'') =',F10.2,
0443 & ' > TBRST =',F10.2)
0444 1850 FORMAT(' EXCEEDED COND ARRAY DIMENSION')
0445 1860 FORMAT('FLIGHT TIME GREATER THAN 110 MIN.')
0446 1866 FORMAT('STOPPED BY OPERATOR')
0447 C
0448 C
0449 C
0450 C
0451 C
0452 90 WRITE(IOUT,1903)
0453 1900 FORMAT(2X, 'CONDENSER DONE.'/' DECOMMUTATOR DONE.'/
0454 & ' INTERPOLATION FOLLOWS......')
0455 C
0456 DO 196 JC = 1,JK
0457
0458 IBC = ICOND(2,JC)
0459 IF(IBC .GT. 999)ICOND(2,JC) = IBC/1000
0460 IF(ICOND(2,JC) .GT. 260)ICOND(2,JC) = 0
0461 INDEX = ICOND(1,JC)
0462 IF(ICOND(1,JC) .GT. 10.AND.ICOND(1,JC).LE.19)ICOND(1,JC) = 1
0463 IF(ICOND(1,JC) .GE.10.AND.ICOND(1,JC).LE.49)ICOND(1,JC) = 4
0464 IF (ICOND(1,JC) .EQ.0.DR.ICOND(1,JC) .GT. 5)ICOND(1,JC) = 5
ORIGINAL FORM OF POOR QUALITY

PAGE 009 MAIN .SA:1 UMET-1 15 DEC 1982

0465
0466 95 FORMAT(1', ' JK', 6X, 'TIME: OF DAY; ELAPSED',
0467 & ' TNNP. TEMP. REF. HIGH REF. REL.HUM. UNDECOM',
0468 & ' BAROSWITCH WORKING WORKING',
0469 & ' /6X,' HH:MM:SS.S HOURS MH:SS.S (SEC'), 21(','),
0470 & ' --(HZ)--', 21(''),', CONTACT # CONTACT CHANNEL',/
0471 & RT2 = (COND(1,JC)+TANCH)/3600.
0472 & RT4 = COND(2,JC)
0473 & ELPT = COND(1,JC)
0474 & RT3 = ELPT/60.
0475 & DUMMY =ELPT/60.
0476 & IDUMMY=IDUMMY : ITRUNCATES VALUE
0477 & DUMMY=DUMMY-DUMMY1 : ITRUNCATED REAL
0478 & RT3=DUMMY*60. : GIVES THE REMAINDER
0479 & IT2 = (RT2-IT1)*60.
0480 & RT2 = (RT2-IT2)*60.
0481 & IT1 = RT1
0482 & IT2 = (RT2-IT1)*60.
0483 & IT3 = ELFT/60.
0484 & ITZ = (COND(I,JC)+TLANCH)/3600.
0485 & RT4 = COND(2,JC)
0486 & RTZ = (COND(I,JC)-TANCH)/3600.

0493 & GO TO (191U, 1920, 1930, 1940, 1950) IDUMMY
0494 & IDUMMY = JC-1
0495 & IF(MOD(IDUMMY,LINE6) . EQ. 0) WRITE(IDOUT,95)
0496 & IDUMMY=IDUMMY: ITRUNCATES VALUE
0497 & IDUMMY=IDUMMY=IDUMMY1 : ITRUNCATED REAL
0498 & RT3=DUMMY*60. : GIVES THE REMAINDER
0499 & IT2 = (RT2-IT1)*60.
0500 & IDUMMY=IDUMMY: ITRUNCATES VALUE
0501 & GO TO 196
0502 & WRITE(IDOUT,192) JC,IT1,IT2,RT1,RT2,IT3,RT3,RT4,COND(3,JC),
0503 & & ICOND(2,JC),IUC,INDAX
0504 & & ICOND(2,JC),IUC,INDAX
0505 & RT2 = (RT2-IT2)*60.
0506 & GO TO 196
0507 & WRITE(IDOUT,193) JC,IT1,IT2,RT1,RT2,IT3,RT3,RT4,COND(3,JC),
0508 & & ICOND(2,JC),IUC,INDAX
0509 & GT TO 196
0510 & WRITE(IDOUT,194) JC,IT1,IT2,RT1,RT2,IT3,RT3,RT4,COND(3,JC),
0511 & & ICOND(2,JC),IUC,INDAX
0512 & & ICOND(2,JC),IUC,INDAX
0513 & GO TO 196
0514 & WRITE(IDOUT,195) JC,IT1,IT2,RT1,RT2,IT3,RT3,RT4,COND(3,JC),
0515 & & ICOND(2,JC),IUC,INDAX
0516 & & ICOND(2,JC),IUC,INDAX
0517 & & ICOND(2,JC),IUC,INDAX
0518 & & ICOND(2,JC),IUC,INDAX
0519 & & ICOND(2,JC),IUC,INDAX
0520 & & ICOND(2,JC),IUC,INDAX
0521 & & ICOND(2,JC),IUC,INDAX
0522 & & ICOND(2,JC),IUC,INDAX

CALL INTERP(JK,TNOH,ISTOP,LCNTU
**PRINT TABLE**

**WRITE(IOUT,1030)**

1030 FORMAT('1',13X,'DECOMMUTATED OUTPUT AT UNIFORM TIME INTERVALS')

**WRITE(IOUT,1131)**

1131 FORMAT(5X,'INDEX TIME AZIMUTH ELEVATION PRESSURE REF FREQ',5X,5X,T,'TEMP REL HUM')

**WRITE(IOUT,1230)**

1230 FORMAT(4X,'(SEC) (DEG) (DEG) (MB) (HZ) (ORDINATES)')

**DO 6767 LL=I,LIST**

6767 CONTINUE

**DO 6767 LL=I,LIST**

6767 CONTINUE

**WRITE(IOUT,1330)IDUMMY,VL(1,LL),VL(2,LL),VL(3,LL),VL(4,LL),VL(5,LL),VL(6,LL),VL(7,LL)**

**WRITE(IOUT,1043)**

1043 FORMAT('xxx REAL TIME PROCESSING COMPLETE *** ')

**WRITE(IOUT,1042)**

1042 KNTCT,LCNTK

**WRITE(IOUT,1045)**

1045 FORMAT('STOPPED, CONDENSER UNABLE TO FIND SIGNAL, TOO NOISY')

**WRITE(IOUT,1046)**

1046 FORMAT('END OF INPUT DATA (EOF)')

**WRITE(IOUT,1047)**

1047 FORMAT('STOPPED, REACH TSTOP (TSTART+TPROC)')

**WRITE(IOUT,1050)**

1050 FORMAT('STOPPED, COND OVERFLOW')

**WRITE(IOUT,1051)**

1051 FORMAT('FLIGHT TIME IN EXCESS OF 120 MIN.')

**WRITE(IOUT,1052)**

1052 FORMAT('STOPPED BY OPERATOR')

**DO 501 LL=1,LIST**

501 J = LIST - LL + 1

**IF(VL(4,J) .GT. 0.0) GO TO 502**

502 CONTINUE

**WRITE(IOUT,6000)**

**WRITE(IOUT,1052)**

1052 CONTINUE

**WRITE PROD COMPATIBLE TAPE**

**DOUBLE CHECK FOR ZERO PRESSURE VALUES AT TOP OF FLIGHT**

**GO TO 502**

**CONTINUE**
LIST = J
CONTINUE
NCDS=LIST ; PASS THIS NUMBER TO PROC
ONE MINUTE DATA IS PASSED IN VL ARRAY
IOUT=102 ; SO PROC OUTPUT GOES TO THE PRINTER
CALL JUMPER
6000 FORMAT(1X,'UNABLE TO FIND NON ZERO PRESSURE - EXPECT ERROR ',
& 'IN ECC-PROC PROGRAM')
STOP
END
0001 OPT L, REL
0002 OPT LLE=80
0003 NAM OFF
0004 IDNT OFF
0005 XDEF OFF
0006 *****
0007 *****
0008 CSCT
0009 IDATA RMB 300
0010 PSCT
0011 OFF LDA #06
0012 STA >*E017
0013 RTS
0014 END

15 DEC 1982
THIS PROGRAM SETS UP THE PRINTER TO OUTPUT 132 CHAR PER LINE

PROGRAM FOLLOWS

PRINTER JSR @$5852 ;INIT PRINTER
LDA @$16
JSR @$565A ;SEND
LDA @$14
JSR @$565A ;PRINTER SET UP FOR 132 CHAR PER LINE
ORA @$30
ANDA @$7F
STA @$E007 ;THIS DISABLES THE SHIFT REGISTER
RTS
END
THIS IS THE INTERRUPT PROGRAM

DATA FROM THE TRADT IS VALID ON THE PIA'S

TO CAUSE AN INTERRUPT, THIS PROGRAM READS THE DATA INTO BYTES WHERE IT IS FORMATTED INTO INTEGERS AND STORED IN IRAMS, WHERE FORTRAN SUB 'FORM' RE-FORMATS THE DATA INTO REAL NUMBERS AND STORES THEM IN TEMP STORAGE. THEN INCREMENTS IFLAG AND RETURNS TO THIS PROGRAM WHERE A RTI IS DONE.

THE COND PROGRAM THEN IS LOOKING FOR IFLAG TO BE SET, SO IT CAN TAKE THE DATA FROM THE TEMP STORAGE AND USE IT.

CSCT

IFLAG RMB 2
BYTE10 RMB 1
BYTE8 RMB 1
BYTE7 RMB 1
BYTE6 RMB 1
BYTE5 RMB 1
BYTE4 RMB 1
BYTE3 RMB 1
BYTE2 RMB 1
BYTE1 RMB 1
BYTE11 RMB 1
BYTE12 RMB 1
BYTE13 RMB 1
BYTE14 RMB 1
BYTE15 RMB 1
BYTE16 RMB 1
TEMPX RMB 2
INTMSK RMB 2
IRAM1 RMB 2
IRAM2 RMB 2
IRAM3 RMB 2
IRAM4 RMB 2
IRAM5 RMB 2
IRAM6 RMB 2
IRAM7 RMB 2
IRAM8 RMB 2
IRAM9 RMB 2
IRAM10 RMB 2
ITOP RMB 2
IDATA RMB 1500

IRAM ARE STACKED INTO THE IDATA ARRAY TO BE PROCESSED

REAL LDU $#810C
LOX $#E000
NOTE HERE THAT PIAS ARE HARD WIRED SO RS0 GOES TO ADDRESS A1 AND RS1 GOES TO A0. THIS PUT THE DATA PORTS NEXT TO EACH IN MEMORY, TO USE OF THE D REGISTER.

;START GETTING THE DATA
0065 PSHU D ;STORES PART OF MET WORD AT BYTEM &BYTED
0066 LDD 9,X ;GETS MORE OF THE MET WORD
0067 PSHU D ;STORES MORE OF THE MET WORD AT BYTEM &BYTED
0068 LDD 8,X
0069 PSHU D
0070 LDD 12,X
0071 PSHU D
0072 LDD 16,X
0073 PSHU D ;IRAM DATA IS NOW STORED FROM BYTE1-BYTE10

;FORMAT THE DATA INTO INTEGER IN IRAM USING BYTE11-BYTE16
0076 AS SCRATCH PAD, THEY ARE STORED AS FOLLOWS:
0077 IRAM1=HOURS
0078 IRAM2=MIN
0079 IRAM3=SEC
0080 IRAM4=.SEC
0081 IRAM5=DEGREES OF EL
0082 IRAM6=DEGREES OF EL
0083 IRAM7=DEGREES OF AZ
0084 IRAM8=DEGREES OF AZ
0085 IRAM9=THOUSANDS PLACES IN MET WORD
0086 IRAM10=LOWEST 3 VALUES OF MET WORD (IE UP TO 999)
0087 DO NOT WRITE IN NEW DATA IF FORTRAN IS FORMATTING OLD DATA
0088 LDD INTMSK
0089 CMFD #$01
0090 BNE DOIT
0091 LDD #$02
0092 STD INTMSK
0093 LDRA DONE
0094 LDIT LDB BYTE10
0095 LSRB
0100 LSRB
0101 STB BYTE11
0102 LSRB
0103 LSRB
0104 LSRB
0105 LSRB
0106 LDA BYTE11
0107 ANDA #$0F
0108 STA BYTE11
0109 LDA #$0A
0110 MUL
0111 ADDB BYTE11
0112 STD IRAM1
0113 NOW DO MIN
0114 LDA BYTE10
0115 ANDA #$03
0116 STA BYTE11
0117 LDA BYTE9
0118 ASLA
0119 ECR NOCAR
0120 LDA BYTE11
0121 LSLA
0122 STA $01
0123 BRA LOWMIN
0124 BRA LOWMIN
0125 NOCAR LDA BYTE11
0126 LSLA
0127 STA BYTE11
0128 LOWMIN LDA BYTE9
0129 ANDA $078
0130 LSRA
0131 LSRA
0132 LSRA
0133 STA BYTE13
0134 LDB BYTE12
0135 LDA $0A
0136 MUL
0137 ADDB BYTE13
0138 STD IRAM2
0139 * NOW DO SECONDS
0140 LDA BYTE8
0141 LSRA
0142 LSRA
0143 LSRA
0144 LSRA
0145 STA BYTE11
0146 LDB BYTE9
0147 ANDB $07
0148 LDA $0A
0149 MUL
0150 ADDB BYTE11
0151 STD IRAM3
0152 * NOW DO TENTHS OF SECONDS
0153 LDB BYTE8
0154 ANDB $0F
0155 LDA $00
0156 STD IRAM4
0157 *
0158 * TIME WORD DONE
0159 *
0160 LDA BYTE7
0161 ANDA $50F
0162 STA BYTE11
0163 LDB BYTE7
0164 ANDB $FF0
0165 LSRE
0166 LSRE
0167 LSRE
0168 LSRE
0169 LDA $0A
0170 MUL
0171 ADDB BYTE11
0172 STD IRAMS
0173 *
0174 LDA BYTE6
PAGE 004 REAL • SA:1 UMET-1 15 DEC 1982

0175 ANDA #$0F
0176 STA BYTE11
0177 LDB BYTE6
0178 ANDB #$F0
0179 LSRB
0180 LSRB
0181 LSRB
0182 LSRB
0183 LDA #$0A
0184 MUL
0185 ADDB BYTE11
0186 STD IRAM6
0187 * DONE WITH EL
0188 * START ON AZ WORD
0189 LDE BYTE5
0190 LSRB
0191 LSRB
0192 LSRB
0193 LSRB
0194 LDA #$69
0195 MUL
0196 STD BYTE13
0197 LDB BYTE5
0198 ANDB #$0F
0199 LDA #$0A
0200 MUL
0201 ADDB BYTE13
0202 STD BYTE13
0203 LDA #$00
0204 LDB BYTE4
0205 LSRB
0206 LSRB
0207 LSRB
0208 LSRB
0209 ADDB BYTE13
0210 STD IRAM7
0211 * NOW DO TENTHS OF DEGREE AZ
0212 LDA BYTE3
0213 LSRA
0214 LSRA
0215 LSRA
0216 LSRA
0217 STA BYTE11
0218 LDB BYTE4
0219 ANDB #$0F
0220 LDA #$0A
0221 MUL
0222 ADDB BYTE11
0223 STD IRAM8
0224 * AZ WORD DONE
0225 * START ON MET WORD
0226 LDA BYTE2
0227 LSRA
0228 LSRA
0229 LSRA
0230 LSRA
0231 STA BYTE11
0232 LDB BYTE3
Of Poor (1 Li;)

PAGE 005 REAL .SAII UMET-1 15 DEC 1982

0233 ANDE #00F
0234 LDA #00A
0235 MUL
0236 ADDS BYTE11
0237 STD IRAMP
0238 DO LAST OF MET WORD
0239 LDD BYTE21
0240 ANDE #00F
0241 LDA #00A
0242 MUL
0243 STD BYTE12
0244 LDD BYTE11
0245 ANDD #7001
0246 STD BYTE11
0247 LDD BYTE11
0248 LDD BYTE13
0249 STD BYTE10
0250 DATA IN FORMATTED AS INTERGERS
0251 NOW FORMAT INTO IDATA ARRY FOR USE IN MAIN
MET PROGRAM.
0252
0253
0254 LDD ITOP
0255 ADDD #001
0256 CMFD #150
0257 ENE ITOP
0258 LDD #00000000
0259 ITOP STD ITOP
0260 CMFD ITOT
0261 ENE SETFLG
0262 LDD ITOT
0263 ADDD #001
0264 CMFD #150
0265 BEG ATTOP
0266 STD ITOT
0267 BRA SETFLG
0268 ATTOP LDD #000000
0269 STD ITOT
0270 SETFLG LDD ITOP
0271 CMFD ITOT
0272 BLO WORKUP
0273 STD IBOT
0274 STD IFLAG
0275 BRA STOR
0276 WORKUP ADDD #150
0277 STD IBOT
0278 STD IFLAG
0279 STD IRAM1
0280 LDD IRAM2
0291 STD ,X++
0292 LOD IRAM3
0293 STD ,X++
0294 LOD IRAM4
0295 STD ,X++
0296 LOD IRAM5
0297 STD ,X++
0298 LOD IRAM6
0299 STD ,X++
0300 LOD IRAM7
0301 STD ,X++
0302 LOD IRAM8
0303 STD ,X++
0304 LOD IRAM9
0305 STD ,X++
0306 LOD IRAM10
0307 STD ,X++
0308 CMFX #&BCE6  ;HIGHEST ADDRESS OF DATA STACK
0309 EMS LOOPFX
0310 STX TEMFX
0311 ERA DONE
0312 LOOPFX LDX #&812E ;BOTTOM ADDRESS OF DATA STACK
0313 STX TEMFX
0314 LOD #&100
0315 STD #TOP
0316 DONE RTI
0317 END
SUBROUTINE SEARCH

SUBROUTINE SEARCH (LOS, JK, MASK)

INCLUDE 'COMMON.SA'

COMMON BLOCK (COMMON.SA)

SEARCH FROM LOW TO HIGH FREQ. (SIGNAL MORE OFTEN LOW)

BND = SIGMIN

COUNTS ( KB, LL, TLL) BELOW MOVING BOUNDS,

CLEANEST POINT COUNT ( KENDG) AND INQX ( IBND)

END = SIGMIN

KBLL = 0

DO 662 IB = 1, IN

KLL = 0

DO 661 J = 1, 10

661 IF (FREQ(J) < LT, BND ) KB = KB + 1

KEND = KB - KBLL

BND = BND + HGATE

IF (KBLL .GE. IBND) GO TO 662

IF (KB .LT. KENDG ) GO TO 662

IBND = IB

KBND = KEND

CONTINUE

10 IF(KENDG .GE. 3) GO TO 669

SIGNAL NOT FOUND • INCREMENT NOISE COUNT

LOS = LOS + 1

IF (LOS .GT. 1000) GOTO 1111

IF (LOS .EQ. 1) TSWCH2 = TIME(1)

GET NEXT FIVE RAW DATA POINTS

RETURN

FOUND SIGNAL • SET GATE • NOTE LEADING

EDGE SWITCH TIME AND Dwell,

CONDENSE THE DATA POINT, AND PROCEED TO DECOMMUTATE.

CONTINUE

LOSN = LOS

LOS = 0

SIGLEV = SIGMIN + (IBND-2)*HGATE
TSWCH1 = TSWTCH
IF(TIME(1) .LT. TSWTCH) GOTO 44
TSWCH = TIME(1)
44 DWELL = (TSWTCH - TSWCH1)

CONDENSER OUTPUT

IF(NSUM .EQ. 0) GO TO 2665
IF(JK .EQ. 0) GOTO 4889
IF(TSWCH1 .LE. COND(1,JK)) GO TO 3004
JK = JK + 1
COND(1,JK) = TSWCH1
COND(2,JK) = DWELL
COND(3,JK) = FSUM/NSUM
IF(JK .GE. 900) GOTO 2222

DECOMMUTATE

IF(JK .GT. 1) GO TO 3001
WRITE(104,3005)
3005 FORMAT('***** COND. MATRIX *****',/)
CONTINUE
WRITE(100,3000) JK,COND(1,JK),COND(2,JK),COND(3,JK)
3000 FORMAT( 'COND(',I3,')=',F12.1,F8.3,F10.4,5X)

GO TO 2665

1111 MASK = 1
RETURN
2222 MASK = 2
RETURN

WRITE(IOUT,3007) JK
3007 FORMAT( ' *** WARNING *** SWITCH TIME DID NOT INCREASE AT',
JH = ',I3)
2665 FSUM = 0.
NSUM = 0
RETURN
END
0001 OPT L.REL
0002 OPT L.LE=80
0003 NAM SETUP
0004 IDNT SETUP
0005 XDEP SETUP
0006 *
0007 ****************** THIS IS THE MAIN PAPER TAPE SETUP PROGRAM ************
0008 *
0009 PSCT
0010 IFLAG RMB 2
0011 BYTE10 RMB 1
0012 BYTE11 RMB 1
0013 BYTE12 RMB 1
0014 BYTE13 RMB 1
0015 BYTE14 RMB 1
0016 BYTE15 RMB 1
0017 BYTE16 RMB 1
0018 BYTE17 RMB 1
0019 BYTE18 RMB 1
0020 BYTE19 RMB 1
0021 BYTE20 RMB 1
0022 IRAM1 RMB 2
0023 IRAM2 RMB 2
0024 IRAM3 RMB 2
0025 IRAM4 RMB 2
0026 IRAM5 RMB 2
0027 IRAM6 RMB 2
0028 IRAM7 RMB 2
0029 IRAM8 RMB 2
0030 IRAM9 RMB 2
0031 IRAM10 RMB 2
0032 IRAM11 RMB 2
0033 IRAM12 RMB 2
0034 IRAM13 RMB 2
0035 IRAM14 RMB 2
0036 IRAM15 RMB 2
0037 IRAM16 RMB 2
0038 IRAM17 RMB 2
0039 IRAM18 RMB 2
0040 IRAM19 RMB 2
0041 IRAM20 RMB 2
0042 IRAM21 RMB 2
0043 IDATA RMB 3000 ;IRAM ARE STACKED INTO THE IDATA ARRY TO BE PROCESSED
0044 ***********************
0045 *
0046 *
0047 * PROGRAM FOLLOWS
0048 *
0049 PSCT
0050 SETUP LDD #0000
0051 STA BYTE1 D.P. FLAG
0052 STA BYTE2 TEMP, STORAGE FOR DATA
0053 STA BYTE3 MULTIPLIER
0054 STA BYTE4 X REGISTER
0055 STA BYTE5
0056 STA BYTE6 1000 FLAG
0057 STA BYTE7 DELETE FLAG
0058 STA BYTE8 STACK COUNTER
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0059</td>
<td>STA BYTE9</td>
<td>ENTRY COUNTER</td>
</tr>
<tr>
<td>0060</td>
<td>STA BYTE10</td>
<td>STORAGE FOR INTEGER</td>
</tr>
<tr>
<td>0061</td>
<td>STA BYTE11</td>
<td>STORAGE FOR FRACTION</td>
</tr>
<tr>
<td>0062</td>
<td>STD IRAM1</td>
<td></td>
</tr>
<tr>
<td>0063</td>
<td>STD IRAM2</td>
<td></td>
</tr>
<tr>
<td>0064</td>
<td>STD IRAM3</td>
<td></td>
</tr>
<tr>
<td>0065</td>
<td>STD IRAM4</td>
<td></td>
</tr>
<tr>
<td>0066</td>
<td>STD IRAM5</td>
<td></td>
</tr>
<tr>
<td>0067</td>
<td>STD IRAM6</td>
<td></td>
</tr>
<tr>
<td>0068</td>
<td>STD IFLAG</td>
<td></td>
</tr>
<tr>
<td>0069</td>
<td>LDA $09</td>
<td></td>
</tr>
<tr>
<td>0070</td>
<td>STA $E017</td>
<td></td>
</tr>
<tr>
<td>0071</td>
<td>LDD $900</td>
<td></td>
</tr>
<tr>
<td>0072</td>
<td>STD IRAM4</td>
<td></td>
</tr>
<tr>
<td>0073</td>
<td>LDD $2878</td>
<td></td>
</tr>
<tr>
<td>0074</td>
<td>STD $DFFE</td>
<td></td>
</tr>
<tr>
<td>0075</td>
<td>LDA $07</td>
<td></td>
</tr>
<tr>
<td>0076</td>
<td>STA $E016</td>
<td></td>
</tr>
<tr>
<td>0077</td>
<td>ANOCC $EF</td>
<td></td>
</tr>
<tr>
<td>0078</td>
<td>RTS</td>
<td></td>
</tr>
<tr>
<td>0079</td>
<td>END</td>
<td></td>
</tr>
<tr>
<td>0080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0081</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
THIS PROGRAM LOOKS TO SEE IF THE 'S' KEY OF THE TERMINAL
HAS BEEN HIT. IF SO THIS PROGRAM SET ISTPER TO 1. THEN ADVANCE
LOOKS AT THIS FLAG, AND IF SET CAUSES THE PROGRAM TO STOP.

CSCT
IFLAG RMB 2
BYTE10 RMB: 1
BYTE9 RMB 1
BYTE8 RMB 1
BYTE7 RMB 1
BYTE6 RMB 1
BYTE5 RMB 1
BYTE4 RMB 1
BYTE3 RMB 1
BYTE2 RMB 1
BYTE1 RMB 1
BYTE11 RMB 1
BYTE12 RMB 1
BYTE13 RMB 1
BYTE14 RMB 1
ISTPER RMB 2
TCMPX RMB 2
INTMSK RMB 2
IRAM1 RMB 2
IRAM2 RMB 2
IRAM3 RMB 2
IRAM4 RMB 2
IRAM5 RMB 2
IRAM6 RMB 2
IRAM7 RMB 2
IRAM8 RMB 2
IRAM9 RMB 2
IRAM10 RMB 2
ITOP RMB 2
IBOT RMB 2
IDATA RMB 1500 ;IRAM ARE STACKED INTO THE IDATA ARRY TO BE PROCESSED
* PROGRAM BEGINS

FSCT
STPER LDD #$0000
STD ISTPER
LDA >SEC14
LSRA
BCC NOCHAR ;IF FLAG NOT SET THEN NO CHAR
LDA >SEC15 ;LOAD THE DATA FROM THE ACIA
ASCII LETTER S
BNE NOCHAR ;IF NOT EQUAL TO S RETURN
LDD #$0001 ;TO SET THE FLAG
STD ISTPER ;FLAG IS NOW SET
0001 OPT L,REL
0002 OPT LLE=80
0003 NAME TAPE1
0004 IDENT TAPE1
0005 *
0006 *********** THIS IS THE MAIN PAPER TAPE PROGRAM ***********
0007 *   DATA FROM THE TAPE READER IS VALID ON THE PIA
0008 *   TO CAUSE AN INTERRUPT, THIS PROGRAM READS THE DATA
0009 *   INTO BYTES WHERE IT IS FORMATTED INTO INTERGERS
0010 *   AND STORED IN IRAMS, WHERE FORTRAN SUB 'FORM' RE-FORMATS
0011 *   THE DATA INTO REAL NUMBERS AND STORES THEN IN THE CAL.
0012 *
0013 *
0014 CSCT
0015 IFLAG RMB 2
0016 BYTE10 RMB 1
0017 BYTE9 RMB 1
0018 BYTE8 RMB 1
0019 BYTE7 RMB 1
0020 BYTE6 RMB 1
0021 BYTE5 RMB 1
0022 BYTE4 RMB 1
0023 BYTE3 RMB 1
0024 BYTE2 RMB 1
0025 BYTE1 RMB 1
0026 BYTE11 RMB 1
0027 BYTE12 RMB 1
0028 BYTE13 RMB 1
0029 BYTE14 RMB 1
0030 BYTE15 RMB 1
0031 BYTE16 RMB 1
0032 BYTE17 RMB 1
0033 BYTE18 RMB 1
0034 BYTE19 RMB 1
0035 BYTE20 RMB 1
0036 IRAM1 RMB 2
0037 IRAM2 RMB 2
0038 IRAM3 RMB 2
0039 IRAM4 RMB 2
0040 IRAM5 RMB 2
0041 IRAM6 RMB 2
0042 IRAM7 RMB 2
0043 IRAM8 RMB 2
0044 IRAM9 RMB 2
0045 IRAM10 RMB 2
0046 ITOP RMB 2
0047 IDOT RMB 2
0048 IDATA RMB 3000 ;IRAM ARE STACKED INTO THE IDATA ARRY TO BE PROCESSED
0049 ********************
0050 *
0051 *
0052 * PROGRAM follows
0053 *
0054 PSCT
0055 STY IRAM9
0056 LDY IRAM6
0057 LDA $E014
0058 LDA $E015 LOAD DATA INTO ACC.
PAGE 002  TAPE1  .SA11  UMET-1  15 DEC 1982

0059  STA  BYTE2
0060  LDB  BYTE7
0061  CMPB  #$05
0062  BEO  SPAC
0063  CMFA  #$7F
0064  BEO  RED
0065  LDB  #$60
0066  STE  BYTE7
0067  BRA  OUT
0068  RED  INC  BYTE7
0069  OUT  STA  BYTE2
0070  STY  IRAM6
0071  LDY  IRAM4
0072  RTI
0073  SPAC  CMFA  #$20
0074  BNE  NSPAC
0075  LDB  #$66
0076  STE  BYTE8
0077  NSPAC  CMFA  #$2E
0078  BNE  NPER
0079  LDB  #$01
0080  STE  BYTE1
0081  BRA  OUT
0082  NPER  CMFA  #$30
0083  BLO  OUT  NUMBER?
0084  CMFA  #$39
0085  BLS  MASK
0086  BRA  OUT
0087  MASK  ANDA  #$0F
0088  STA  -Y
0089  LDB  BYTE8
0090  DECB
0091  STE  BYTE8
0092  LOA  BYTE18
0093  CMFA  #$120
0094  BLO  NZER
0095  LDA  BYTE2
0096  ANDA  #$0F
0097  CMFA  #$00
0098  BNE  NZER
0099  LDB  BYTE8
0100  CMFA  #$05
0101  BEO  ZER
0102  NZER  LDB  BYTE1
0103  CMFA  #$01
0104  BEO  RES
0105  BRA  OUT
0106  RES  LDE  #$00
0107  STE  BYTE1
0108  ZER  LDD  #$0000
0109  STD  IRAM1
0110  STD  IRAM2
0111  LDB  ,Y+  X=X0  ACCUMULATOR
0112  INC  BYTE8
0113  STD  IRAM2
0114  LDB  #$01
0115  MULT  STE  BYTE3  MULTIPLIER = X
0116  LDA  BYTE8

THREE CONSECUTIVE DELETES?
RESET DELETE FLAG
COUNTER SET TO SIX
SET D.P. FLAG
NUMBER?
STORE ON STACK
120 ENTRIES?
ACCEPTS ZEROS AFTER 160 ENTRIES.
FIRST DIGIT? (ACCEPTING ZEROS)
CHECK D.P. FLAG
RESET D.P. FLAG
INITIALIZE TO ZERO
X=X0  ACCUMULATOR
0117 CMPA #$05    LAST DIGIT?
0118 BEQ LAST    FINISHED WITH ENTRY?
0119 CMPA #$06    1000'S PLACE?
0120 BEQ FINI
0121 CMPA #$02    SET 1000'S FLAG
0122 ENE THOU
0123 LDA BYTE6    THOU LDA ,Y+
0124 INC BYTE6    INC BYTE6
0125 THOU LDA ,Y+
0126 INC BYTE6    MULTIPLY BY POWER OF TEN
0127 MUL
0128 ADDD IRAM1   MUL
0129 STD IRAM1
0130 LDA BYTE6
0131 LDE #$10
0132 MUL          INCREMENT MULTIPLIER
0133 BRA MUL
0134 LAST LDA BYTE6
0135 CMPA #$91
0136 ENE THOU
0137 DEC BYTE6    RESET 1000'S FLAG
0138 LDA ,Y+
0139 INC BYTE6
0140 LDE #$6A    TEN
0141 MUL          TIMES BY TEN
0142 LDA #$44    100
0143 MUL          TIMES BY 100
0144 ADDD IRAM1
0145 STD IRAM1
0146 FINI LDE #$0001 SET FLAG
0147 STD IFLAG
0148 INC BYTE18   STA BYTE6
0149 LDA #$06    RESET COUNTER
0150 STA BYTE6
0151 JMP OUT
0152 END
SUBROUTINE TRACK(LOS, JK, MASK)

COMMON BOLD

INCLUDE 'COMMON.SA'

SET GATE BOUNDS
COUNT SIGNAL POINTS IN GATE

MASK = 0
IF (BUFR.GT.SIGMAX) BUFR = SIGMAX
BLWR = SIGLEV - HGATE
IF (BLWR.LT.SIGMIN) BLWR = SIGMIN

MEMORY TO STABILIZE CONDENSED SIGNAL

NGATE = 1
SUMGTE = SIGLEV

COUNT POINTS IN GATE
DO 671 J = 1, 10
IF (FREG(J).GT.BUFR .OR. FREG(J).LT.BLWR) GO TO 671
SUMGTE = SUMGTE + FREG(J)
NGATE = NGATE + 1
IF (NGATE.GT.6) GO TO 672
671 CONTINUE

IF LESS THAN TWO (EXCLUDING SIGLEV) IN GATE, LOST SIGNAL
IF (NGATE.GT.2) GO TO 672
CALL SEARCH(LOS, JK, MASK)
IF (MASK.EQ.1) GO TO 63
IF (MASK.EQ.2) GO TO 85
RETURN

HAVE SIGNAL # INCREMENT FOR MEAN AND ADJUST GATE
672 FSUM = FSUM + SUMGTE/NGATE
NSUM = NSUM + 1
SIGLEV = (SIGLEV + SUMGTE / NGATE) * 0.5
RETURN

CONTINUE
63 CONTINUE
85 CONTINUE
83 CONTINUE

LOST SIGNAL --- NEVER FOUND IT IN SEARCH

END

JK > 1000 (COND DIMENSION)
PAGE 002  TRACK  .SA:1  UMET-1  15 DEC 1982
0059    MASK=2
0060    RETURN
0061
0062    END
FUNCTION ADIR(U,V)

THIS SUBROUTINE WILL COMPUTE THE DIRECTION FROM WHICH THE WIND IS BLOWING.

DIMENSION IFF ACTUAL ARGUMENTS ARE SUBSCRIPTED.

DUMMY=V/U

IF (U) 61, 63

ADIR = 90. - 57.2958* ATAN(DUMMY)
GO TO 69

ADIR = 270. - 57.2958* ATAN(DUMMY)
GO TO 69

IF (V) 65, 66, 67

ADIR = 360.
GO TO 69

ADIR = 999.
GO TO 69

ADIR = 180.
RETURN
END
C FUNCTION TO TAKE MOD OF REAL

FUNCTION AMOD(A,E)

DUMMY=A/E
IDUM=DUMMY
DUMMY2=IDUM
AMOD=(DUMMY-DUMMY2)*E
RETURN
END
0001 C FUNCTION TO TAKE BASE 10 LOG OF A NUMBER
0002 FUNCTION ALOG10(A)
0003
0004 ALOG10=(ALOG(A))/(ALOG(10.))
0005 RETURN
0006 END
C FUNCTION TO TURN INTEGER TO REAL NUMBER

FUNCTION FLOAT(I)

DUMMY=I
FLOAT=DUMMY
RETURN
END
ROUTINE FOR REDUCTION OF ONE MINUTE DATA

SUBROUTINE PRO

INCLUDE 'PRDCOM.SA'

DIMENSION IIERR(0), OLD(5), IERRM(488), AKKMB(36) ; TEST IERRM & AKKMB
DIMENSION EPR(110), ETP(110), EPT(110), NLET(6) ; NLET FOR TEST
DIMENSION F'(110), P3(110), TEMPF'(110), DLEL(3) ; DLEL FOR TEST

DATA EI/0.0/
DATA AKY.ME/;

D L0QU .0 x9011 .0 ,850.0,800.0,500.0,400.0,350.0,300.0,250.0,200.0,175.0,150.0,125.0,100.0,80.0,60.0,50.0,40.0,35.0,30.0,1/0.0/

CONSTANTS USED IN PROGRAM

DATA ZERO/0.0/, IZERO/0/, ONE/1.0/, IDNE/1/, TEN/10.0/ ; TESTING
DATA ISEVEN/7/, SEVEN/7.0/, ZNINE/-99.99/, Z180/180.0/ ; TESTING
DATA ZNN/-999.99/, TWO/2.0/, POINT5/0.5/, ZTENF'2/10.2/ ; TESTING
DATA Z99/99.0/, Z98F'9/90.9/, Z56/56.0/, Z113/113.0/, Z273/273.15/, Z59/59.0/, Z58/58.0/, Z57/57.15/, Z56/56.0/, Z55/55.0/, Z54/54.0/, Z53/53.15/, Z52/52.0/, Z51/51.0/, Z50/50.0/, Z49/49.0/, Z48/48.0/, Z47/47.15/, Z46/46.0/, Z45/45.15/, Z44/44.0/, Z43/43.15/, Z42/42.0/, Z41/41.0/, Z40/40.0/, Z39/39.0/, Z38/38.0/, Z37/37.15/, Z36/36.0/, Z35/35.15/, Z34/34.0/, Z33/33.15/, Z32/32.0/, Z31/31.0/, Z30/30.0/, Z29/29.0/, Z28/28.0/, Z27/27.15/, Z26/26.0/, Z25/25.15/, Z24/24.0/, Z23/23.15/, Z22/22.0/, Z21/21.0/, Z20/20.0/, Z19/19.0/, Z18/18.0/, Z17/17.15/, Z16/16.0/, Z15/15.15/, Z14/14.0/, Z13/13.0/, Z12/12.0/, Z11/11.0/, Z10/10.0/, Z9/9.0/, Z8/8.0/, Z7/7.0/, Z6/6.0/, Z5/5.0/, Z4/4.0/, Z3/3.0/, Z2/2.0/, Z1/1.0/, Z/0.0/

FORMATS

100 FORMAT(1X,2A5,2I7,5F6.1,II,2F5.1,F7.1,A6)
101 FORMAT(F6.1,II,F6.1,F8.5,F7.1,F6.2,F7.1,F7.4,F6.1,F7.1,F7.2,F6.1,
6F7.1,F6.1,F7.1,F6.1,F7.4,F6.1,F7.1,F7.2,F6.1,
6F7.1,F6.1,F7.1,F6.1,F7.4,F6.1,F7.1,F7.2,F6.1)
110 FORMAT(1IX,'TIME ALT PRESS ',
120 FORMAT('STATION',1I4,4X,'LAUNCH DATE',1I3,12X,'I2',12X,'I2',
130 FORMAT('LAUNCH TIME',1I2,'',120X,'F3.1,' GMT SONDE ID ',I5,1X)

RECORDED INSTRUMENT HUMIDITIES

""
I

**PAGE 002 PRO**

**UMET-1**

15 DEC 1982

0059 & '*** LESS THAN 20 FRONT NOT LISTED ***')
0060 5000 FORMAT(/'*** GMD IN LIMITING ANGLES ***',/)
0061 & '*** DURING PART OF OBSERVATION ***')
0062 6011 FORMAT((F12.5,A10,2X,Z2,A29,ZX))
0063 8150 FORMAT('xxx ORDER GREATER THAN 100.0 xxxxxx'/
0064 & 'xxx DURING PART OF OBSERVATION xxx')
0065 & 'xxx GMD TRACKING ERROR ***' /
0066 8160 FORMAT(/'xxx GMD IN LIMITING ANGLES xxx',/
0067 9034 FORMAT(//'xxx ',A61)
0068 & 'xxx ORDER GREATER THAN 100.0 xxxxxx' /
0069 & 'xxx DURING PART OF OBSERVATION xxx')
0070 9264 FORMAT(/'xxx GMD TRACKING ERROR ***' /
0071 9364 FORMAT(//'xxx ORDER GREATER THAN 100.0 xxxxxx' /
0072 9464 FORMAT(//'xxx DURING PART OF OBSERVATION xxx')
0073 & 'xxx GMD IN LIMITING ANGLES ***' '
0074 & '*** DURING PART OF OBSERVATION ***')
0075 & '*** DURING PART OF OBSERVATION ***')
0076 & '*** DURING PART OF OBSERVATION ***')
0077 & '*** DURING PART OF OBSERVATION ***')
0078 & '*** DURING PART OF OBSERVATION ***')
0079 & '*** DURING PART OF OBSERVATION ***')
0080 & '*** DURING PART OF OBSERVATION ***')
0081 & '*** DURING PART OF OBSERVATION ***')
0082 & '*** DURING PART OF OBSERVATION ***')
0083 & '*** DURING PART OF OBSERVATION ***')
0084 & '*** DURING PART OF OBSERVATION ***')
0085 & '*** DURING PART OF OBSERVATION ***')
0086 & '*** DURING PART OF OBSERVATION ***')
0087 & '*** DURING PART OF OBSERVATION ***')
0088 & '*** DURING PART OF OBSERVATION ***')
0089 & '*** DURING PART OF OBSERVATION ***')
0090 & '*** DURING PART OF OBSERVATION ***')
0091 & '*** DURING PART OF OBSERVATION ***')
0092 & '*** DURING PART OF OBSERVATION ***')
0093 & '*** DURING PART OF OBSERVATION ***')
0094 & '*** DURING PART OF OBSERVATION ***')
0095 & '*** DURING PART OF OBSERVATION ***')
0096 & '*** DURING PART OF OBSERVATION ***')
0097 & '*** DURING PART OF OBSERVATION ***')
0098 & '*** DURING PART OF OBSERVATION ***')
0099 & '*** DURING PART OF OBSERVATION ***')
0100 & '*** DURING PART OF OBSERVATION ***')
0101 & '*** DURING PART OF OBSERVATION ***')
0102 & '*** DURING PART OF OBSERVATION ***')
0103 & '*** DURING PART OF OBSERVATION ***')
0104 & '*** DURING PART OF OBSERVATION ***')
0105 & '*** DURING PART OF OBSERVATION ***')
0106 & '*** DURING PART OF OBSERVATION ***')
0107 & '*** DURING PART OF OBSERVATION ***')
0108 & '*** DURING PART OF OBSERVATION ***')
0109 & '*** DURING PART OF OBSERVATION ***')
0110 & '*** DURING PART OF OBSERVATION ***')
0111 & '*** DURING PART OF OBSERVATION ***')
0112 & '*** DURING PART OF OBSERVATION ***')
0113 & '*** DURING PART OF OBSERVATION ***')
0114 & '*** DURING PART OF OBSERVATION ***')
0115 & '*** DURING PART OF OBSERVATION ***')
0116 & '*** DURING PART OF OBSERVATION ***')
CassSSSZmzzSzzizzzzzzizs
CassSSSZmzzSzzizzzzzzizs
CassSSSZmzzSzzizzzzzzizs

0117 C ************************************
0118 I=IZERO
0119 DO 50 J=1,7
0120 I=I+1
0121 50 IIERR(I)=0
0122 C
0123 C COMPUTE CALIBRATION CONSTANTS FOR RADIOSONDE DATA
0124 C
0125 CALL TEMPL(CALT,P,IERRF,I,ITEMP)
0126 CALL RL(RECHR,CALT,P,I,IDF,IH)
0127 C
0128 C READ HEADER CARD
0129 C
0130 IF(NCDS .GE. 10B)NCDS = 10B
0131 C **** PROCOMMON ALLOWS ONLY 10B ONE MINUTE ENTRIES
0132 C **** WHERE COND PASSES UP TO 110 MINUTES OF DATA
0133 IZR=0
0134 IERST=NCDS-1
0135 ITERM=NCDS-1
0136 DUMMY=PR(NCDS+1)
0137 DUMMY = ONE
0138 C
0139 C READ DATA CARD
0140 C
0141 DO 577 I=1,NCDS
0142 C
0143 * DATA IS PASSED FROM MET TO PRD IN THE VL ARRAY
0144 C
0145 TIM(I)=VL(1,I)/60.0
0146 PR(I)=VL(4,I)
0147 DT(I)=VL(6,I)
0148 OF(I)=VL(7,I)
0149 AZ(I)=VL(2,I)
0150 EL(I)=VL(3,I)
0151 577 CONTINUE
0152 C
0153 HMGDM=HEIGHT
0154 C
0155 C CHECK INPUT DATA FOR ERRORS
0156 C
0157 DO 40 I=1,NCDS
0158 T(I)=ZERO
0159 TALP(I)=ZERO
0160 C
0161 DEMP(I)=ZERO
0162 IF(TIM(I) .LE. ITME) IIERR(2)=1
0163 IF(I .GT. NCDS) IIERR(3)=1
0164 ITME=TIM(I)
0165 IF(I .EQ. 1) GO TO 10
0166 IF(EL(I) .GT. ZERO .AND. EL(I) .LT. SEVEN) IELVFL=1
0167 IF(AZ(I) .NE. ZERO .AND. EL(I) .GE. SEVEN) GO TO 10
0168 AZ(I)=Z180
0169 EL(I)=Z180
0170 GO TO 14
0171 10 IF(IAZ .EQ. 0) GO TO 14
0172 IF(AZ(I) .LE. Z180) GO TO 12
0173 AZ(I)=AZ(I)-Z180
0174 GO TO 14
0175  12 AZ(I)=AZ(I)+2190
0176  14 IF(PR(I) .NE. 0) GO TO 15
0177  IXT=IXT+1
0178  GO TO 18
0179  15 IF(PR(I) .LE. PRESS .AND. PR(I) .GT. PR(I+1)) GO TO 17
0180  OLD(I)=PR(I)
0181  IERR(4)=1
0182  17 PRESS=PR(I)
0183  TALP(I)=ALOG(PRESS)/2.302585  ;2.302585=LN 10
0184  IF(IXT .EQ. 0) GO TO 18
0185  IHI=I
0186  ILO=IHI-IXT-1
0187  DO 1800 JJ=I,IXT
0189  Y.=JJ+ILO
0190  DUMMY=IHI-ILO
0192  9189 DUMMY = TALP(ILO)+DUMMY/DUMMY1*(TALP(IHI)-TALP(ILO))
0193  9190 DUMMY = TALP(I)
0194  PR(I)=POWER(TALP(I),DUMMY)
0195  IXT=I
0197  13 IF(I .GE. NCDS-1) GO TO 36
0198  DO 1900 JJ=I,IXT
0199  D1=ABS(DUMMY)
0200  D2=ABS(DUMMY1)
0201  IF(D1 .GE. D2) IERR(8)=1
0202  OLD(5)=TIM(I+1)
0203  36 CONTINUE
0204 C
0205 C  PRINT ERROR MESSAGES
0206 C
0207 DO 38 II=1,8
0208 IF(IERR(II) .EQ. 0) GO TO 38
0209 IERR(II)=0
0210 WRITE(IOUT,1115)
0212  1115 FORMAT('DATA CARD ID DOES NOT MATCH FLIGHT ID ***')
0213  2224 IF(I .GT. 2) GOTO 3334
0214 WRITE(IOUT,2225)
0215  2225 FORMAT(' TIME NOT INCREASING - PROGRAM CONTINUES ***')
0216  3334 IF(I .GT. 3) GOTO 4444
0217 WRITE(IOUT,3335)
0218  3335 FORMAT(' TIME GREATER THAN 120 MIN - PROGRAM CONTINUES')
0219  4444 IF(I .GT. 4) GOTO 5554
0220 WRITE(IOUT,4445)
0221  4445 FORMAT('PRESSURE NOT DECREASING - PRESSURE=',F10.4)
0222  5554 IF(I .LT. 5) GOTO 36
0223 WRITE(IOUT,5555)
0224  5555 FORMAT('TEMP ORD FAILS DIFFERENCE TEST ** CHECK TEMP ORD AT ',F10.4)
0225  36 CONTINUE
0226 C
0227 C
0228 IF(DT(I) .GT. DUMMY .OR. OF(I) .GT. DUMMY)
0229 IF(DT(I) .LT. DUMMY .OR. OF(I) .LT. DUMMY)
0230 WRITE(IOUT,8150) TIM(I),PR(I),DT(I),OF(I),AZ(I),EL(I)
0231 IF(AZ(I) .GT. 360.0 .OR. EL(I) .GT. 90)
0232 WRITE(IOUT,8160) TIM(I),PR(I),DT(I),OF(I),AZ(I),EL(I),AZ(I),EL(I)
PROCESS DATA POINTS

DO 90 I=1,NCDS
   PE=ONE
   I=FR(I)
   J=I

CALL TEMF'CE(CALTF',IERF',J,ITEMP)

CALL RL(RECRH,CALTF',J,IDF,IN)

DUMMY=(1000)/PR(I)
DUMMY1=POHER(DUMMY,.28571428)
POTTP(I)=T(I)*DUMMY1

COMPUTE ALTITUDE

IF (I .GT. 1) GO TO 62
DUMMY=PCAL/PR(I)
HGF'(I)=HTM-L+Z14F'6xALOG(DUMMY)*(T(I)+SORT+CENT)
GO TO 11
62 DUMMY=F'RIMEP/F'R(T)
DLOGP=ALOG(DUMMY)
HGF(I)=PRIMEH+Z14F'6xDLOGP*(TV(I)+F'RIMEV)
FRIMEP=FR(I)
PRIMEH=HGF(I)
PRIMEV=TV(I)
IF (DF(I) .EQ. ZERO) DF(I)=Z99P9
IF (IDF .NE. 1) GO TO 83
F(I)=EI
DEHP(I)=EI
CONTINUE

CHECK LAPSE RATE AND POTENTIAL TEMPERATURE

EFR(I)=ZNN
ETP(I)=ZNN
EFT(I)=ZNN
IF (DF(I) .EQ. ZERO) GO TO 90
IF (I .EQ. 1) GO TO 90
DUMMY=HGF(I-1)-HGF(I)
RATE=((T(I)-T(I-1))x(TEN**3))/ABS(DUMMY)
IF (RATE .LT. 2TEMP2) GO TO 84
ETP(I)=RATE
EFR(I)=FR(I)
IF (POTTP(I) .GT. POTTP(I-1)) GO TO 90
EFT(I)=POTTP(I)
0291 EFR(I)*PR(I)
0292 C
0293 90 CONTINUE
0294 C
0295 C
0296 SURTK=SURT+CENT
0297 IF(IDNZFL.EQ.0.AND.FR(NCDS).LE.Z20P) GO TO 700
0298 SUMR=EI
0299 C0300 700 ISN=AMOD(LANCH,100)
0301 WRITE(IOUT,120)IDSTAT,IMONTH,IDAY,IYEAR,I1,I2,TS3,IDSND
0302 WRITE(IOUT,110)FCAL,CALF,RECTF,SURTK,LALRH,RECRH,SURH
0303 WRITE(IOUT,109)
0304 IF(SUMR .EQ. EI) SUMR=Z9F9X5 ;Z9F9X5=9.99999
0305 C
0306 C WINDS COMPUTATION
0307 C
0308 MAX=NCDS
0309 C0310 CONTINUE
0311 IF(PR(I) .LT. HRRH) HRRH=PR(I)
0312 C0313 IF(I .EQ. 1) GO TO 9663
0314 C0315 ANG=(DIR(I)-Z180)*Z1P7E
0316 WNS=SPD(I)*ZCOS(ANG)
0317 WEW=SPD(I)*ZSIN(ANG)
0318 GO TO 9663
0321 IF(AZ(IM1) .EQ. DUMMY .OR. AZ(IP1) .EQ. DUMMY) GO TO 863
0322 ANG=(DIR(I)-Z180)*Z1P7E
0323 SPD=SPD(I)*ZCOS(ANG)
0324 WNS=WNS+SPD(I)*ZCOS(ANG)
0325 9663 CONTINUE
0326 GO TO 9663
0327 C0328 863 DIR(I)=Z99F99 ;Z99F99=999.9
0329 SPD(I)=Z99F99
0330 WNS=Z99F99
0331 WEW=Z99F99
0332 C0333 9663 QQ=(TV(I)/T(I)-ONE)/Z611
0334 C0335 IF(F(I) .EQ. EI) QQ=EI
0336 C0337 9663 C INTERPOLATE WIND VALUES
0338 C
0349 IF(WNS .NE.Z999P9 .AND. WEN .NE.Z999P9) GO TO B663
0350 IF(I .EQ. MAX .OR. I .EQ. 2) GO TO B663
0351 IF(AZ(IM2) .EQ. ZNINE) GO TO B663
0352 ANG=(DIR(IM1)-Z180)*Z1F7E
0353 WNSBD=SFD(IM1)*ZCOS(ANG)
0354 WEN99=SFD(IM1)*ZSIN(ANG)
0355 ANG=DIR(IF'1)-Z180)*Z1F7E
0356 WNS99=SFD(IF'1)*ZCOS(ANG)
0357 WEN99=SFD(IF'1)*ZSIN(ANG)
0358 WNS=WNS99-(WNSBD-WNS99)/TWO
0359 WEN=WEN99-(WENBD-WEN99)/TWO
0340 B663 IF(I .GT. 1) GO TO 610.
0351 WEWSLST=WNS
0352 WNSLST=WNS
0353 C INTERPOLATE STANDARD MET ALTITUDES
0354 C
0355 610 IF(PR(I) .GE. AKYMB(WWK')) GO TO 9963
0356 DUMMY=PR(IM1)
0360 DUMM1=PR(I)
0361 DUMM2=AKYMB(HHKK)
0362 A=(ALOG10(DUMMY)-ALOG10(DUMM2))
0363 E=ALOG10(DUMMY)-ALOG10(DUMM2)
0364 C=A/E
0365 AA99=TIM(IM1)+C(TIM(I)-TIM(IM1))
0366 AA99=TIM(IM1)+C(TIM(I)-TIM(IM1))
0367 B=HGF(IM1)+C(HGF(I)-HGF(IM1))
0368 CC99=EI
0369 DD99=EI
0370 EE99=EI
0371 FF99=EI
0372 IF(PR(I) .GE. AKYMB(WWK')) GO TO 9963
0373 IF(IF(IM1) .GE. IF(I) .AND. IF(I) .NE. IF(IM1)) GO TO 6663
0374 IF(IF(IM1) .GE. IF(I) .AND. IF(I) .NE. IF(IM1)) GO TO 6663
0375 IF(IF(IF'1) .GE. IF(I) .AND. IF(I) .NE. IF(IF'1)) GO TO 6663
0376 IF(IF(IF'1) .GE. IF(I) .AND. IF(I) .NE. IF(IF'1)) GO TO 6663
0377 IF(IF(IF'1) .GE. IF(I) .AND. IF(I) .NE. IF(IF'1)) GO TO 6663
0378 IF(IF(IF'1) .GE. IF(I) .AND. IF(I) .NE. IF(IF'1)) GO TO 6663
0379 8662 CONTINUE
0380 FF99=AKYMB(HHKK)
0381 GG99=T(IM1)+C(T(I)-T(IM1))
0382 HH99=F(IM1)+C(F(I)-F(IM1))
0383 XXX99=DEWPT(I)+C(DEWPT(I)-DEWPT(IM1))
0384 IF(IF(IM1) .GE. IF(I) .AND. IF(I) .NE. IF(IM1)) GO TO 6663
0385 IF(IF(IM1) .GE. IF(I) .AND. IF(I) .NE. IF(IM1)) GO TO 6663
0386 IF(IF(IM1) .GE. IF(I) .AND. IF(I) .NE. IF(IM1)) GO TO 6663
0387 IF(IF(IM1) .GE. IF(I) .AND. IF(I) .NE. IF(IM1)) GO TO 6663
0388 IF(IF(IM1) .GE. IF(I) .AND. IF(I) .NE. IF(IM1)) GO TO 6663
0389 SS99=Z999P9
0390 TT99=Z999P9
0391 VV99=Z999P9
0392 IF(I .EQ. MAX .OR. I .EQ. 2) GO TO 6663
0393 IF(I .EQ. 1 .AND. WNS .NE.Z999P9) GO TO 866
0394 IF(I .EQ. MAX .OR. I .EQ. 2) GO TO 6663
0395 IF(AZ(IM2) .EQ. ZNINE .OR. AZ(IM1) .EQ. ZNINE) GO TO 6663
0396 &
0397 866 AMD=DIR(IM1)-Z180)*Z1F7E
0398 WNS99=SFD(IM1)*ZCOS(ANG)
0399 WEN99=SFD(IM1)*ZSIN(ANG)
0400 UU99=WNSBD-WNSBD-WNS99)
0401 VV99=WEH99+C(WEN-WEH99)
0402 DUMMY=VV99*VV99+UU99*UU99
0403 C
0404 DUMMY=VV99/UU99
0405 TT99=TAN(DUMMY)*Z257
0406 IF(VV99 .LT.ZERO .AND. UU99 .LT.ZERO) TT99=TT99+Z180
0407 IF(VV99 .GT. ZERO .AND. UU99 .GT. ZERO) TT99=TT99+Z1800
0409 IF(VV99 .GT. ZERO .AND. UU99 .LT. ZERO) TT99=TT99+(2*Z180)
0410 WW99=TV(IM1)+Cx(TV(I)-TV(IM1))
0411 XX99=(WW99/GG99-ONE)/ZP611
0412 IF(MH99 .LE. 81) XX99=81-1
0413 YY99=ZERO
0414 IF(I .GT. 2) GO TO 600
0415 IF(WEWLST .EQ.Z999P9 .OR. WNSLST .EQ.Z999P9) GO TO 600
0416 IF(WW99 .LT.ZERO .OR. WNS .LT.ZERO) GO TO 600
0417 UU99=WNSLST+Cx(WNS-WNSLST)
0418 VV99=WEWLST+Cx(WEW-WEWLST)
0419 DUMMY=VV99+UU99
0420 6663 WW99=TV(IM1)+Cx(TO(I)-TV(IM1))
0422 XX99=(WW99/GG99-ONE)/ZP611
0423 IF(MH99 .LE. 81) XX99=81-1
0424 YF99=ZERO
13,414 IF(I .GT. 2) GO TO 600
0425 IF(VV99 .GT.ZERO .AND. UU99 .LT.ZERO) TT99=TT99+(2*Z180)
0426 IF(VV99 .LT.ZERO .AND. UU99 .GT.ZERO) TT99=TT99+Z180
0427 IF(VV99 .LT.ZERO .AND. UU99 .LT.ZERO) TT99=TT99+(2*Z180)
0428 IF(T(I).LE.LZ33F'1 .OR. F(I).EQ.BI .OR. F(I).GT.220P) GO TO 900
0429 IHUMFL=0
0430 F(I)=81
0431 DEWPT(I)=E:I
0432 HH99=61
0433 0099=E:I
0434 XX99=81
0435 900 IF(TV(I) .LE. ZNN) GO TO 800
0436 T(I)=0.
0437 TV(T)=0.
0438 POTTP(I)=0.
0439 HGP(I)=0.
0440 GG99=0.
0441 QQ99=0.
0442 DD99=0.
0443 EE99=0.
0444 HH99=0.
0445 C WRITE INTERPOLATIONS
0447 C 800 AA99P = AA99 + .05
0449 B99P = B99 + .5
0450 FF99P = FF99 + .05 ;ROUNDED FOR PRINTING
0451 PP99P = PP99 + .0005
0452 QQ99P = QQ99 + .05
0453 DD99P = DD99 + .05
0454 HH99P = HH99 + .05
0455 QQ99P = QQ99 + .05
0456 TT99P = TT99 + .05
0457 XX99P = XX99 + .0005
0458 SS99P = SS99 + .05
0459 UU99P = UU99 + .05
0460 VV99P = VV99 + .05
0464 KKKK=KKKK+1
$9963$ IF($P(I) .EQ.AKRMEX(KKSX)) KKSX=KKSX+1
0466 9967 IHUMFL=0
0467 9968 $F(I)=EI$
0468 9969 DEMPT(I)=EI
0469 9970 QD=EI
0470 9971 910 IF($TV(I) .NE. ZNN$) GO TO 810 $;ZNN=-999.9$
0471 9972 $T(I)=EI$
0472 9973 $TV(I)=EI$
0473 9974 POTTP(I)=EI
0474 9975 MCF(I)=EI
0475 9976 810 CONTINUE
0476 9977 IF($SPD(I) .NE.Z999P9 .AND. DIR(I) .NE.Z999P9$) GO TO 811
0477 9978 WNS=Z999P9
0478 9979 WEWP=Z999P9
0479 9980 811 CONTINUE
0480 99C C
0481 99C WRITE PROCESSED DATA
0482 99C $\star$
0483 99C 9984 $TIP=TIM(I)+.05 ;DUMMY VARIABLE FOR PRINTING IE ROUNDS NUMBER$
0484 9985 $HGP=HGP(I)+.5$
0485 9986 $FRF=FRF(I)+.05$
0486 9987 $TLPF=TLP(I)+.00005$
0487 9988 $TPFP=T(I)+.05$
0488 9989 $POTTP=POTTP(I)+.05$
0489 9990 $TUP=TV(I)+.005$
0490 9991 $PFPP=F(I)+.05$
0491 9992 $DEWTP=DEWTP(I)+.05$
0492 9993 $GDF=QD+.00005$
0493 9994 $SPDF=SPD(I)+.05$
0494 9995 $DIRF=DIR(I)+.05$
0495 9996 $WNSF=WNS+.05$
0496 9997 $WEMP=WEWP+.05$
0497 9998 WRITE(IOUT,1010) $TIP,HGPF,FRPF,TLPFP,TPFP,POTTP,TUP,PFPP,DEWTP,GDF,SPDF,DIRF,WNSF,WEMP$
0498 9999 $\star$
0500 999K WRITE(IOUT,1010) TIMF,HGPF,
0501 999K $\star$
0502 999K $\star$
0503 999K WRITE MESSAGES
0504 999C $\star$
0505 999C WRITE(IOUT,9154)
0506 999C $\star$
0507 999C IF($IHUMFL .EQ. 0$) WRITE(IOUT,4000)
0508 999C IF($IELVFL .EQ. 1$) WRITE(IOUT,5000)
0509 999C WRITE(IOUT,9154)
0510 999C IPTFLG=1
0511 999C DO 64 $I=1,NCDS$
0512 999C IF($FRP(I) .LT. ZERO) GO TO 64$
0513 999C IPTFLG=1
0514 999C IF($ETP(I) .GE. -999.$) WRITE(IOUT,9264) EFRP(I),ETF(I) $;ZNNN=-999.999$
0515 999C IF($ETF(I) .GE. -999.$) WRITE(IOUT,9264) EFRP(I),ETF(I)
0516 999C 64 CONTINUE
0517 999C IF($IPTFLG .GT. 0$) WRITE(IOUT,9464)
0518 999C $\star$
0519 999C $\star$
0520 999C 1 RETURN
0521 999C END
COMMON DATA FOR PRO PROGRAM

COMMON TALP(100),AZ(100),DEWPT(100),DIR(100),EL(100),F(100)
COMMON HGPT(100),JUNK(123),VL(7,110),SECS,DUMMY,DUMMY1,IDIUM
COMMON IDUMM1,AMINU,ISTAT,IDSOND,IMONTH,IDAY,ITYEAR,HEIGHT
COMMON FF0,FTEMP0,FR0,RECTP,CALR,NCRN,CPRT,NCDS,IOUT
COMMON VSFC,DSFC
COMMON SF(100),T(100),TIM(100),TV(100),XX(100)
COMMON YY(100),OT(100),OF(100),FR(100)
COMMON FOTT(100)
OPTION STACK=300,USTACK=200

PROGRAM MAIN

C 'PRD' RADIOSonde DATA REDUCTION PROGRAM

C

COMMON DATA FOR PRD PROGRAM

INCLUDE 'PRDCOM.SA'

C

C--- DATA IS PASSED TO PRD FROM MET IN COMMONS

WRITE(100,100)

D010  FORMAT(' AT THE START OF THE PRD PROGRAM')

D020  CALL PRD

C

C

C

C

C

C

CALL PRD

IDUM = 1

! SET UP INFINITE DO LOOP

IDUM = IDUM + 1

IDUM = IDUM - 1

GOTO 300

STOP

END
SUBROUTINE TO DETERMINE RELATIVE HUMIDITY

SUBROUTINE RL(RECRH, CALTF, I, IDF, IH)

**include 'procom.sa'

CENT=273.16
RAD=57.2957795
IF(IH.ne.0) GO TO 4
CALRT=CALTF+CENT
A3=1.5*(RECRH-50.0)/RAD
DUMMY=ZSIN(A3)/ZCOS(A3)
DUMMY1=0.0355*CALT-15.3315
DUMMY2=DUMMY+33.75-0.067*CALT+DUMMY1*ALOG10(CALT)
FK=DUMMY2/(0.08*ALOG10(CALT)+0.27-0.062*CALT)
IH=1
RETURN

CHECK VALIDITY OF RH ORDINATE
4 IF (OF(I) .EQ. 0.) GO TO 8
IF (T(I) .LT. 233.15) GO TO 5
43 IF (MOD(ICK6R,2) .GT. 0) GO TO 46
RH ELEMENT 476 (CARBON)
DF=6.0-OF(I)
CEE=64.8-CALRH/2.51
R=100.0-CALRH
AA=1.66+DF/33.0
GO TO 36
35 CEE=CALRH/1.45-5.198
R=10.0-CALRH
AA=8.5-DF/10.0
36 DUMMY=DF/CEE
DUMMY1=ABS(DUMMY)
DUMMY2=POWER(DUMMY1,AA)

RH ELEMENT 41B (LITHIUM)
A3=0.02617994*(OF(I)-50.0)
DUMMY=((ZSIN(A3)/ZCOS(A3)+33.75-0.087*T(I)-(0.27-0.002*T(I)))*FK
/(15.3315+0.08*FK-0.0355*T(I)))
F(I)=POWER(10.0,DUMMY)
6 IF (F(I) .LT. 0.0) GO TO 8
IF (F(I) .LT. 99.0) GO TO 55
DEWPT(I)=T(I)
IF (F(I) .LT. 100.0) GO TO 55
F(I)=100.0
55 IDF=0
55 DUMMY=(7.5*T(I)-2948.7)/(T(I)-35.9)
DUMMY1=POWER(10.0,DUMMY)
E=F(I)*0.0611*DUMMY1
55 IF (F(I) .EQ. 100.0) GO TO 7
DEWPT(I)=(237.3*ALOG10(E)-186.527)/(8.286-ALOG10(E))
DEWPT(I)=273.15+DEWPT(I)
7 TV(I)=T(I)/(1.0-0.379*PR(I))
0059     GO TO 9
0060     8 TV(I)=T(I)
0061     DEWFP(I)=0.0
0062     F(I)=0.0
0063     IDF=1
0064     RETURN
0065     END
CTEMPCE SUBROUTINE TO CALCULATE TEMPERATURES
SUBROUTINE CTEMPCE(CALTP,IERRF,I.ITEMP)
C SUBROUTINE TO COMPUTE RADIOSONDE TEMPERATURES
C
INCLUDE 'PROCOM.SA'
IERRF=0
CENT=273.16
IF(DT(I) .EQ. 0 .AND. TEMP .NE. 0) GO TO 10
IF(ITEMP .NE. 0) GO TO 53
DUMMY=2.0*RECTP
DUMMY1=POWER(DUMMY,.996626)
R30=-48000.0+8960200.0/DUMMY1
CALTP=CALTP+CENT
TLAST=CENT
DUMMY=1510.0/CALTP
DUMMY1=POWER(CALTP,2.95)
AINV=RECTP*EXP(DUMMY1)/(DUMMY1*(95.0-RECTP))
ITEMP=1
RETURN
53 IF(ICREIN .GT. 1) GO TO 60
C TEMPERATURE ELEMENT
FACTOR=AINV*(95.0/DT(I)-1.0)
DUMMY=1510.0/TLAST
EXPT=EXP(DUMMY)
DUMMY=POWER(1510.0/EXPT,2.95)
DUMMY1=POWER(1510.0/TLAST,2.95)
T(I)=TLAST+(EXPT-FACTOR/DUMMY)/(1510.0*EXPT/(TLAST*TLAST)+2.95)
DUMMY=TLAST-T(I)
IF(Wi(DUMMY) .LE. 0.02) GO TO 54
S
T(I)=T(I)-CENT
T(I)-CENT+T(I)-T(I)
FACTOR=AINV*(95.0/DT(I)-1.0)
DUMMY=1510.0/TLAST
EXPT=EXP(DUMMY)
DUMMY=POWER(1510.0/EXPT,2.95)
DUMMY1=POWER(1510.0/TLAST,2.95)
T(I)=TLAST+(EXPT-FACTOR/DUMMY)/(1510.0*EXPT/(TLAST*TLAST)+2.95)
DUMMY=TLAST-T(I)
IF(Wi(DUMMY) .LE. 0.02) GO TO 54
S
T(I)=T(I)-CENT
T(I)=CENT+T(I)-T(I)*(1.0+T(I)*T(I)*T(I))/9150625.0+(17.0+CALTP)
R30=17.0+CALTP
R30=17.0+CALTP
54 GO TO 30
C TEMPERATURE ELEMENT
DUMMY=2.0*DT(I)
DUMMY1=POWER(DUMMY,.996626)
REST=-48000.0+8960200.0/DUMMY1
DUMMY=REST/R30
DUMMY2=ALOG(DUMMY)
DUMMY3=ALOG(10.)
FACTOR=DUMMY2/DUMMY3
T(I)=1.0/(3.298588E-3+1.07701E-3*FACTOR+5.89986E-5*(FACTOR*2))
RETURN
C NO TEMPERATURE INPUT
T(I)=-999.99
RETURN
END
CREDITABLE DOCUMENT
OF POOR QUALITY

PAGE 901 WINDS .SA:11 UMET-1 15 DEC 1982

0001 C WINDS SUBROUTINE TO CALCULATE WINDS
0002 C
0003 SUBROUTINE WINDS(MAX,HGMDT,.DISSFC,.MIN)
0004 **********************************************
0005 C
0006 C INCLUDE 'PRDCOM.SA'
0007 C
0008 C DIMENSION V(125),U(125)
0009 C RAD=57.29578
0010 C E=6371.
0011 C SFDC(1)=VSFC
0012 C DIR(1)=DSFC
0013 C ANG=DSFC/RAD
0014 C V(1)=ZCOS(ANG)*SFDC(1)
0015 C U(1)=ZSIN(ANG)*SFDC(1)
0016 C C=16.66667
0017 C DISP=DISSFC/1000.
0018 C DUMMY=AZ(1)/RAD
0019 C XX(1)=DISP*ZCOS(DUMMY)
0020 C YY(1)=DISP*ZSIN(DUMMY)
0021 C ISTART=2
0022 C IF(MIN .NE. 1) ISTART=MIN
0023 C DO 10 M=ISTART,MAX
0024 C HEIGH=HGP(M)-HGMDT
0025 C Z=0.001*ABS(HEIGH)
0026 C AZR=AZ(M)/RAD
0027 C IF(EL(M) .GE. 0.0) EL(M)=0.0001
0028 C DUMMY1=EL(M)
0029 C DUMMY=ABS(DUMMY1)
0030 C ELR=EL(M)/RAD
0031 C DUMMY=1.0+Z/R/ZCOS(ELR)
0032 C DUMMY1=DUMMY*2-1.0
0033 C DUMMY2=SQRT(DUMMY1)
0034 C DUMMY3=ATAN(DUMMY2)
0035 C DISP=R*(DUMMY3-ELR)
0036 C XX(M)=DISP*ZCOS(AZR)
0037 C YY(M)=DISP*ZSIN(AZR)
0038 10 CONTINUE
0039 C MM2=MAX-2
0040 C MM1=MAX-1
0041 C MPI=MIN+1
0042 C MP2=MIN+2
0043 C U(MPI)=C(XX(MPI)-XX(MIN))/(TIM(MPI)-TIM(MIN))
0044 C V(MPI)=C(YY(MPI)-YY(MIN))/(TIM(MPI)-TIM(MIN))
0045 C U(MPI)=C(XX(MAX)-XX(MMIN))/(TIM(MAX)-TIM(MMIN))
0046 C V(MMI)=C(YY(MMAX)-YY(MMIN))/(TIM(MAX)-TIM(MMIN))
0047 C DO 20 M=MP2,MM2
0048 C IF(EL(M) .GT. 10.0) GO TO 21
0049 C INF=M+2
0050 C INM=M-2
0051 C GO TO 22
0052 21 INF=M+1
0053 C INM=M-1
0054 22 DUMMY=TIM(INF)-TIM(INM)
0055 C U(M)=C(XX(INF)-XX(INM))/DUMMY
0056 20 V(M)=C(YY(INF)-YY(INM))/DUMMY
0057 C DO 30 M=MPI,MM1
0058 C DUMMY=U(M)*U(M)+V(M)*V(M)
0059 C
0659  SPD(M)=SDRT(DUMMY)
0660  DUMMY=U(M)
0661  DUMMY1=V(M)
0662  20 DIR(M)=ADIR(DUMMY,DUMMY1)
0663  SPD(MAX)=SPD(MH1)
0664  DIR(MAX)=DIR(MH1)
0665  RETURN
0666  END
FUNCTION ZCOS(A)

THE 6809 CANT TAKE COS OF A NEG

DUMMY=ABS(A)
ZCOS=COS(DUMMY)
RETURN
FUNCTION ZSIN(A)
C
THE 6809 WILL NOT TAKE THE SIN OF A NEG 0
C
THIS FUNCTION TAKES CARE OF THE PROBLEM
C
IF(A .GE. 0.0) GOTO 10
A = -A
ZSIN = SIN(A)
ZSIN = -ZSIN
RETURN
ZSIN = SIN(A)
RETURN
APPENDIX B

Sample Output

The output under the operational printer option (printer output code = 1) is given here as an example. No adjustments of the algorithm have been made to optimize decommutation or other performance, nor is there an evaluation offered here of the adequacy of the adapted software.
OF POOR QUALITY

-----------

******* INPUT DATA *******

STATION ID = 72
STATION HEIGHT = 4.4
ZERO AZIMUTH = 1
DATE = 3-11-92
SURFACE TEMP = 66.1
SURFACE RH = 66.1
SURFACE PRESS = 1012.8
SURFACE WIND SPEED = 4.0
SURFACE WIND DIRECTION = 65.0
Sonde ID = 9987
REFERENCE ORD = 92.0
TEMP CAL ORD = 73.9
RH CAL ORD = 58.3
SURFACE TIAR ORD = 61.6
SURFACE RH ORD = 44.8
Sonde TYPE = 2
BARST CONTACT = 100.0
<table>
<thead>
<tr>
<th></th>
<th>1817.8</th>
<th>1844.2</th>
<th>1829.0</th>
<th>1823.6</th>
<th>1812.4</th>
<th>1811.2</th>
<th>999.8</th>
<th>978.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>937.8</td>
<td>937.0</td>
<td>946.8</td>
<td>925.0</td>
<td>921.4</td>
<td>913.4</td>
<td>943.4</td>
<td>892.4</td>
</tr>
<tr>
<td>2</td>
<td>821.6</td>
<td>871.2</td>
<td>864.8</td>
<td>858.2</td>
<td>859.4</td>
<td>829.4</td>
<td>819.9</td>
<td>808.4</td>
</tr>
<tr>
<td>3</td>
<td>718.4</td>
<td>766.2</td>
<td>778.2</td>
<td>748.2</td>
<td>757.8</td>
<td>740.2</td>
<td>739.7</td>
<td>733.2</td>
</tr>
<tr>
<td>4</td>
<td>617.2</td>
<td>673.2</td>
<td>623.8</td>
<td>614.4</td>
<td>645.4</td>
<td>615.4</td>
<td>594.4</td>
<td>579.4</td>
</tr>
<tr>
<td>5</td>
<td>517.0</td>
<td>561.4</td>
<td>592.1</td>
<td>592.8</td>
<td>593.4</td>
<td>521.8</td>
<td>520.1</td>
<td>518.4</td>
</tr>
<tr>
<td>6</td>
<td>417.0</td>
<td>453.8</td>
<td>465.2</td>
<td>477.2</td>
<td>469.4</td>
<td>441.2</td>
<td>453.4</td>
<td>445.6</td>
</tr>
<tr>
<td>7</td>
<td>317.0</td>
<td>422.6</td>
<td>415.8</td>
<td>407.0</td>
<td>404.0</td>
<td>392.0</td>
<td>385.8</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>217.0</td>
<td>321.4</td>
<td>364.2</td>
<td>357.4</td>
<td>350.6</td>
<td>343.0</td>
<td>337.4</td>
<td>336.6</td>
</tr>
<tr>
<td>9</td>
<td>117.0</td>
<td>217.4</td>
<td>318.8</td>
<td>344.0</td>
<td>296.6</td>
<td>292.4</td>
<td>286.2</td>
<td>281.2</td>
</tr>
<tr>
<td>10</td>
<td>97.2</td>
<td>248.4</td>
<td>262.0</td>
<td>256.0</td>
<td>251.2</td>
<td>245.6</td>
<td>240.2</td>
<td>234.8</td>
</tr>
<tr>
<td>11</td>
<td>97.2</td>
<td>224.2</td>
<td>219.0</td>
<td>214.3</td>
<td>248.0</td>
<td>243.0</td>
<td>199.2</td>
<td>194.2</td>
</tr>
<tr>
<td>12</td>
<td>77.2</td>
<td>119.6</td>
<td>169.4</td>
<td>175.0</td>
<td>171.4</td>
<td>167.0</td>
<td>162.0</td>
<td>158.6</td>
</tr>
<tr>
<td>13</td>
<td>57.4</td>
<td>105.0</td>
<td>131.4</td>
<td>145.4</td>
<td>142.4</td>
<td>138.0</td>
<td>130.6</td>
<td>127.0</td>
</tr>
<tr>
<td>14</td>
<td>37.2</td>
<td>121.8</td>
<td>166.6</td>
<td>152.0</td>
<td>149.0</td>
<td>145.2</td>
<td>141.8</td>
<td>139.0</td>
</tr>
<tr>
<td>15</td>
<td>27.2</td>
<td>49.4</td>
<td>67.4</td>
<td>57.4</td>
<td>44.6</td>
<td>41.8</td>
<td>39.0</td>
<td>37.4</td>
</tr>
<tr>
<td>16</td>
<td>27.2</td>
<td>31.2</td>
<td>27.0</td>
<td>25.2</td>
<td>22.6</td>
<td>20.4</td>
<td>17.6</td>
<td>15.8</td>
</tr>
<tr>
<td>17</td>
<td>17.2</td>
<td>9.2</td>
<td>4.6</td>
<td>3.6</td>
<td>2.6</td>
<td>2.0</td>
<td>1.4</td>
<td>1.0</td>
</tr>
<tr>
<td>18</td>
<td>6.8</td>
<td>4.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

| EFFECTIVE CONTACT NUMBER AT LAUNCH  | 3.47 |
| HIGHEST CONTACT NUMBER CALIBRATED  | 149 |

BALLOON RELEASED AT 14472120.0

| TEMP O/G FAILS DIFFERENCE TEST || CHECK TEMP O/G AT  | 1.0000 |
| TEMP O/G FAILS DIFFERENCE TEST || CHECK TEMP O/G AT  | 2.0000 |

ORIGINAL PAGE IS OF POOR QUALITY
### Surface Conditions

<table>
<thead>
<tr>
<th>TIME</th>
<th>ALT</th>
<th>PRESS</th>
<th>LOG</th>
<th>TEMP</th>
<th>DTRM</th>
<th>VTEMP</th>
<th>HUMTTY</th>
<th>DECMF</th>
<th>SPECIF</th>
<th>SPD</th>
<th>DUR</th>
<th>HS</th>
<th>GA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSL</td>
<td>FT</td>
<td>IN</td>
<td>FT</td>
<td>FT</td>
<td>FT</td>
<td>FT</td>
<td>FT</td>
<td>FT</td>
<td>FT</td>
<td>FT</td>
<td>FT</td>
<td>FT</td>
</tr>
<tr>
<td>4:10</td>
<td>1183.7</td>
<td>3.4144</td>
<td>271.1</td>
<td>274.5</td>
<td>277.1</td>
<td>65.4</td>
<td>271.2</td>
<td>1.1822</td>
<td>4.4</td>
<td>65.6</td>
<td>-1.6</td>
<td>-3.6</td>
<td></td>
</tr>
<tr>
<td>4:20</td>
<td>1086.6</td>
<td>3.4015</td>
<td>275.2</td>
<td>275.2</td>
<td>275.2</td>
<td>64.7</td>
<td>276.6</td>
<td>1.1829</td>
<td>399.9</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td></td>
</tr>
<tr>
<td>4:30</td>
<td>997.4</td>
<td>3.3897</td>
<td>275.0</td>
<td>275.2</td>
<td>275.2</td>
<td>66.0</td>
<td>269.5</td>
<td>1.1879</td>
<td>399.9</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td></td>
</tr>
<tr>
<td>4:40</td>
<td>906.3</td>
<td>3.3763</td>
<td>276.1</td>
<td>277.1</td>
<td>277.1</td>
<td>8.0</td>
<td>1.0</td>
<td>1000</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td></td>
</tr>
<tr>
<td>4:50</td>
<td>815.9</td>
<td>3.3640</td>
<td>274.3</td>
<td>275.2</td>
<td>275.2</td>
<td>35.1</td>
<td>264.8</td>
<td>1.1716</td>
<td>399.9</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td></td>
</tr>
<tr>
<td>5:00</td>
<td>724.0</td>
<td>3.3512</td>
<td>274.2</td>
<td>274.2</td>
<td>274.2</td>
<td>49.9</td>
<td>264.7</td>
<td>1.1827</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td></td>
</tr>
<tr>
<td>5:10</td>
<td>632.3</td>
<td>3.3387</td>
<td>272.2</td>
<td>274.5</td>
<td>274.5</td>
<td>49.9</td>
<td>276.7</td>
<td>1.1827</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td></td>
</tr>
<tr>
<td>5:20</td>
<td>540.9</td>
<td>3.3263</td>
<td>272.2</td>
<td>274.5</td>
<td>274.5</td>
<td>49.9</td>
<td>276.7</td>
<td>1.1827</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td></td>
</tr>
<tr>
<td>5:30</td>
<td>449.4</td>
<td>3.3140</td>
<td>271.2</td>
<td>274.5</td>
<td>274.5</td>
<td>49.9</td>
<td>276.7</td>
<td>1.1827</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td></td>
</tr>
<tr>
<td>5:40</td>
<td>358.0</td>
<td>3.3018</td>
<td>270.2</td>
<td>274.5</td>
<td>274.5</td>
<td>49.9</td>
<td>276.7</td>
<td>1.1827</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td></td>
</tr>
<tr>
<td>5:50</td>
<td>266.6</td>
<td>3.2897</td>
<td>269.2</td>
<td>274.5</td>
<td>274.5</td>
<td>49.9</td>
<td>276.7</td>
<td>1.1827</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td></td>
</tr>
<tr>
<td>6:00</td>
<td>175.1</td>
<td>3.2778</td>
<td>268.2</td>
<td>274.5</td>
<td>274.5</td>
<td>49.9</td>
<td>276.7</td>
<td>1.1827</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td></td>
</tr>
<tr>
<td>6:10</td>
<td>83.6</td>
<td>3.2660</td>
<td>267.2</td>
<td>274.5</td>
<td>274.5</td>
<td>49.9</td>
<td>276.7</td>
<td>1.1827</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td></td>
</tr>
<tr>
<td>6:20</td>
<td>32.1</td>
<td>3.2543</td>
<td>266.2</td>
<td>274.5</td>
<td>274.5</td>
<td>49.9</td>
<td>276.7</td>
<td>1.1827</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td></td>
</tr>
<tr>
<td>6:30</td>
<td>0.6</td>
<td>3.2427</td>
<td>265.2</td>
<td>274.5</td>
<td>274.5</td>
<td>49.9</td>
<td>276.7</td>
<td>1.1827</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td></td>
</tr>
<tr>
<td>6:40</td>
<td>0.6</td>
<td>3.2314</td>
<td>264.2</td>
<td>274.5</td>
<td>274.5</td>
<td>49.9</td>
<td>276.7</td>
<td>1.1827</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td></td>
</tr>
<tr>
<td>6:50</td>
<td>0.6</td>
<td>3.2202</td>
<td>263.2</td>
<td>274.5</td>
<td>274.5</td>
<td>49.9</td>
<td>276.7</td>
<td>1.1827</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td>999.9</td>
<td></td>
</tr>
</tbody>
</table>

### Additional Information

- **Press Differential:** No significant changes in pressure differential observed.
- **Temperature:** Temperatures show a gradual decrease throughout.
- **Relative Humidity:** Relatively low, maintaining consistent values.
- **Wind Speed:** Steady wind conditions with minor fluctuations.
- **Visibility:** Good visibility with occasional light haze.

Overall, the conditions remain stable with minor variations, indicating a calm and predictable atmosphere.
RECORDED INSTRUMENT HUMIDITIES

LESS THAN 20 PRCT NOT LISTED

END IN LIMITING ANGLES

DURING PART OF OBSERVATION

LAYER BELOW 836.5 MB HAS SUPER ADIABATIC LAPSE RATE OF 13.4 DEG/MI

LEVEL BELOW 836.5 MB

POTENTIAL TEMPERATURE = 287.9 DEG K

NOT INCREASING

LAYER BELOW 134.7 MB HAS SUPER ADIABATIC LAPSE RATE OF 111.4 DEG/MI

LEVEL BELOW 134.7 MB

POTENTIAL TEMPERATURE = 461.8 DEG K

NOT INCREASING

LAYER BELOW 131.1 MB HAS SUPER ADIABATIC LAPSE RATE OF 103.1 DEG/MI

LEVEL BELOW 131.1 MB

POTENTIAL TEMPERATURE = 462.0 DEG K

NOT INCREASING

LAYER BELOW 125.7 MB HAS SUPER ADIABATIC LAPSE RATE OF 49.4 DEG/MI

LEVEL BELOW 125.7 MB

POTENTIAL TEMPERATURE = 388.2 DEG K

NOT INCREASING

FOR THE ABOVE LAYERS OR LEVELS, CHECK TEMP ORDINATE AND PRESSURE ENTRIES.

NORMAL PROGRAM TERMINATION
APPENDIX C

Circuit Diagrams
Fig. C-1. Paper tape reader and front panel cable C3.
Fig. C-2. Biphasic-to-level converter.
Fig. C-3. Input board — paper tape interface.
Fig. C-4. Input board layout.
APPENDIX D

Future Application to Rocketsonde Data: METROC-R

The following is a listing of partially completed software for the real-time processing of rocket (DATASONDE) meteorological data. The routines are those of METROC-K,* adapted to the portable microprocessor-based UMET system. METROC-R is the current operational version of the University of Utah computer program for automatic processing, at NASA Wallops Flight Center, of digitally recorded rocket-launched parachute-sonde data.

All routines below have compiled successfully for the microprocessor system, except DRIVER, which is not yet completely adapted, and MAIN, which has not yet been treated. Remaining work includes the modification of UMET-1 real-time hardware and software to include the slant range word, and programming to accept and process keyboard inputs for the rocketsonde system and to produce the desired output hardcopy format. Memory management design must be completed with efficient use of the available RAM. The two-page program concept, wherein real-time (AMET) processing is followed by "derived data" and output processing (BMET), adequately accommodates rocket meteorological data processing requirements.

The processing of DATASONDE data is considerably less difficult than that for the RAWINSONDE because the need for decommutation is essentially absent. "Tie on" to a RAWINSONDE pressure level is re-

quired, however, if, in a future system design in which a portable tracking system and sonde-borne transponder is used and a sonde-borne pressure switch is added, a quite accurate, reliable, and self-contained real-time system concept could be realized.
Fig. C-5. Interface board.
Fig. C-6. Interface board layout. Only the custom-installed components are shown; for a description of the rest of the board, see [10].

- A1: Paper tape shift register 74164
- B1: Chip select logic 7408
- C1: Interrupt decode logic 7411
- A2 - A7, B2 - B7, C2: Real-time data shift registers 74164
- PIA1 - PIA6: Peripheral Interface Adapter 68B21
Subroutine Diagram of the METROC program
PURPOSE

This system of routines performs operations on the double queue shown below. The two queues "chase each other's tails".

ADD1 adds single data points to queue1.
ADD2 adds single data points to queue2.
REMOV1 removes single data points from queue1.
REMOV2 removes single data points from queue2.
REWRT replaces the frequency datum in an interval of points at the head of queue1 with an "inert" value.

Each 'data point' or 'place' in the queue contains 5 values. All data is added and removed as 5-vectors.

---

 queue 2

```
 DDDDDDDDDDD_________E EEEEEEEEEEE
```

 queue 1

```
 EEEE EEEEEEE
```

(values from DRIVER) DDDDDDD_________E EEEEEEEEEEE

 ITAIL2 IHEAD1

DUMMY VARIABLES

D(5) Data vector to be transferred between QFILE and calling routine
EMTY Empty queue flag. Values are:
   1 = queue is empty; no point retrieved
   -1 = not empty
FULL Full queue flag. Values are:
   1 = this queue is full; cannot add point
   -1 = not full

INTERNAL VARIABLES

IPASS Flag to indicate the first call to ADD1 so pointers can be initialized. Values:
   1 = first pass; initialize pointers
   0 = subsequent pass; don't initialize

COMMON VARIABLES

IHEAD1 Head pointer for queue1. Points to location of next point to be added to queue1.
IHEAD2 Head pointer for queue2. Points to location of next point to be added to queue2.
ITAIL1 Tail pointer for queue1. Points to location of next point to be removed from queue1.
ITAIL2 Tail pointer for queue2. Points to location of next point to be removed from queue2.
LNCHTM Length of queue = second dimension of QFILE
QFILE(5,1000) Storage queue containing edited data points from EDIT and meteorological values from DRIVER.
QFILE(1,j) = time since launch (seconds)
QFILE(2,j) = frequency datum (Hertz)
QFILE(3,j) = X coordinate (meters)
QFILE(4,j) = Y coordinate (meters)
QFILE(5,j) = Z coordinate (meters)
CALLING ROUTINES - EDIT, DRIVER

COMMENTS
1. Subroutine ADD1 initializes the queue pointers used by all the other queue access routines; ADD2, REMOVI, REMOVI2, REWRT. Therefore ADD1 must be called at least once before to be called.
2. LNGTHQ must be changed when QFILE is redimensioned.
3. To set up queue for use in METROC:
   IHEAD1=ITAIL1=(headstart)
   IHEAD2=ITAIL2=1
   where (headstart) is just large enough to prevent queue2 from catching up to queue1 in DRIVER. Finding an adequate headstart may require repeated trials with real flight data, so the program may need modification to allow the initial values to be changed at run time.

CHANGES
ADD1 is a new routine and was not part of METROC-K.

COMPILED 6/20/82

REFERENCES

SUBROUTINE ADD1 (D,FULL)
INCLUDE 'RCOMMON.SA1'
DIMENSION D(5)
DATA IFPASS/ 1/
IF (IFPASS.EQ.1) THEN
   IHEAD1=10
   ITAIL1=10
   IHEAD2=1
   ITAIL2=1
   LNGTHQ=2000
   IFPASS=0
   END IF
   FULL=-1.
   IF (IHEAD1.EQ.ITAIL2) THEN ;if head has reached tail
      FULL=1.
     ;of queue2, queue1 is full
      RETURN
   END IF
   DO 10 I=1,5
      QFILE(I,IHEAD1)=D(I)
10 CONTINUE
   IHEAD1=IHEAD1+1
   IF (IHEAD1.GT.LNGTHQ) IHEAD1=1
END
SUBROUTINE ADD2 (D,FULL)
C------------------------------------
C For documentation see ADD1
C------------------------------------
INCLUDE 'RCOMMON.SA:1'
DIMENSION D(5)
FULL=-1.
IF (IHEAD2.EQ.ITAL1) THEN
   FULL=1.
   RETURN
END IF
DO 10 I=1,5
   QFILE(I,IHEAD2)=D(I)
10 CONTINUE
IHEAD2=IHEAD2+1
IF (IHEAD2.LT.LENGTHQ) IHEAD2=1
END
SUBROUTINE ADVANC (JJ,TSTOP,TLANCH,TGMDAQ,MASK)

COMMON BLOCK

INCLUDE 'COMMON.SA'

ADVANCE, 5 NEW RAW DATA POINTS.

MASK=0

DO 2 JI=1,5
  JS= JI+5
  TIME(JI)=TIME(JS)
  FREQ(JI)=FREQ(JS)
  AZ(JI)=AZ(JS)
  EL(JI)=EL(JS)
2 CONTINUE

HANDLE REAL TIME DATA

DO 3 J=6,10
  IF(IFLAG .GT. 0)GOTO 3224
  CALL STPER
  IF(ISTPER .NE. 1)GOTO 3223 ;IF STOP IS HIT ISTPER=1
  MASK=4
  RETURN
3224 INTMSK=1
  IF(IBOT .GE. 150)INTMSK=0 ;IBOT POINTS TO OLDEST NOT YET PROCESSED POI
  IDUM=IBOT*10
  IBOT=IBOT+1 ;THE IDATA ARRAY IS ONLY 75 POINTS LONG
  IFLAG=IFLAG-1
  DUMMY=IDATA(IDUM+9)
  DUMMY1=IDATA(IDUM+10)
  FREQ(J)=(DUMMY*1000)+DUMMY1
  DUMMY=IDATA(IDUM+6)
  DUMMY1=IDATA(IDUM+5)
  EL(J)=DUMMY1+(DUMMY*.01)
  DUMMY=IDATA(IDUM+8)
  DUMMY1=IDATA(IDUM+7)
  AZ(J)=DUMMY1+(DUMMY*.01)
  DUMMY=IDATA(IDUM+4)
  DUMMY1=IDATA(IDUM+3)
  SECS=DUMMY1+(DUMMY*.1)
  DUMMY=IDATA(IDUM+2)
  AMINU=DUMMY*60.+SECS
  HOURS=(IDATA(IDUM+1))x3600.
  TIME(J)=HOURS+AMINU-TLANCH ;TIME IS IN SECONDS FROM LAUNCH

IF(INTMSK .EQ. 1)GOTO 444

CALL GETBCK
444 INTMSK=0

IF(TIME(J) .LT. TIME(J-1))GOTO 5
  DUMMY=TIME(J-1)+60.
  IF(TIME(J) .GT. DUMMY)GOTO 5
5 CONTINUE

REAL TIME DATA NOW IN PROGRAM
0059  IF(EL(J) .LT. 0.1) EL(J) = .1
0060  IF(AZ(J) .GE. 360.0) AZ(J) = .1
0061  IF(AZ(J) .LT. 0.1) AZ(J) = .1
0062  IF(FREQ(J) .LT. 1.0) GOTO 5 ; REAL DATA IS SOMETIMES 0.0
0063  FREQ(J) = 1/(FREQ(J) * .000001) ; DATA COMES IN IN USEC
0064  *
0065  IF(FREQ(J) .LT. 4.8 .OR. FREQ(J) .GT. 200.) GOTO 5
0066  GO TO 3
0067  5  J = J - 1
0068  3  CONTINUE
0069  IF(TIME(9) .GT. TSTOP) GOTO 100
0070  *
0071  *
0072  *
0073  51  IF(TIME(4) .LT. VL(1, LIST)) GOTO 53
0074  IF(TIME(4) .LT. TGMDAQ) GOTO 52  
0075  *********************** CALL ANGLE ***********************
0076  CALL ANGLE
0077  VL(2, LIST) = AZ(5)
0078  VL(3, LIST) = EL(5)
0079  52  LIST = LIST + 1
0080  IF(LIST .GE. 120) GOTO 128 ; VL TABLE IS FULL
0081  VL(1, LIST) = VL(1, LIST - 1) + DLIST
0082  GO TO 51
0083  53  CONTINUE
0084  RETURN
0085  *
0086  82  CONTINUE
0087  C
0088  C  END OF DATA
0089  C
0090  MASK = 2
0091  RETURN
0092  C
0093  100  MASK = 1
0094  RETURN
0095  C
0096  128  MASK = 3
0097  RETURN
0098  END
0099
C 9/28/82  CHRIS BURKS
C
C PURPOSE:  
'CLOBSET' MODIFIES THE TCBEAD AND RCSOND  
ARRAYS FOR LATER USE BY THE SUBROUTINE 'FRQTMP'.  
IT TAKES THE LOGARITHM OF RCSOND AND CONVERTS  
TCBEAD FROM CENTIGRADE TO KELVIN.  

LOCAL VARIABLES:  
  I LOOP INDEX  
  SAV TEMPORARY STORAGE  

COMMON VARIABLES:  
  NCBEAD NUMBER OF THERMISTOR CALIBRATION POINTS  
  NCSOND NUMBER OF SONDE CALIBRATION POINTS  
  RCSOND(25) SONDE CALIBRATION DATA  
  TCBEAD(10) THERMISTOR CALIBRATION DATA  

CHANGES MADE:  
1.PUT ALL COMMON VARIABLES INTO FILE 'RCOMMON.SA'  
   AS DESCRIBED IN 'MDOS FORTRAN REFERENCE MANUAL', P.1-6  
2.ADDED 'SAV' VARIABLE BECAUSE FUNCTIONS  
   WILL NOT ACCEPT SUBSCRIPTED VARIABLE ARGUMENTS  
3.ADDED THIS DOCUMENTATION  

MODIFICATION IS FINISHED  
WILL COMPARE  
HAS NOT BEEN RUN  

SUBROUTINE CLOBSET  
INCLUDE 'RCOMMON.SA:1'  
DO 10 I = 1,NCSOND  
   SAV = RCSOND(I)  
10 RCSOND(I) = ALOG(SAV)  
DO 20 I = 1,NCBEAD  
20 TCBEAD(I) = TCBEAD(I) + 273.16  
RETURN  
END
SUBROUTINE DIRSIT (IDRSIT, NDRST, LTHF1, LTHF2, LTHT, TIN, XIN, TOUT, XOUT, VOUT, AOUT, ISLIDE, ISWITCH)

C Provides smoothed sonde positions and temperatures as well as their first and second derivatives (velocities and accelerations).

DUMMY VARIABLES

IDIRSIT A switch to indicate when smoothed data is available to DRIVER and to control initialization. Values:
1 = initialize coefficients in smoothing equations.
2 = add raw data point to time and position arrays (raw data) in POSFIT.
3 = add raw data point to time, position, and temperature arrays.
4 = add points to all above arrays and process all derivatives; return smoothed data. (all arrays are filled - enough points available to start processing)

NDRST Switch to indicate whether temperature data will be processed. Possible values:
3 = wind-only data run; therefore only position data will be smoothed
4 = both position and temperature data will be smoothed.

LTHP1 Smoothing interval length to obtain velocity in POSFIT.
LTHP2 Smoothing interval length to obtain acceleration POSFIT.
LTHT Smoothing interval length for temperature and its first and second derivative.

TIN Time of raw data point.
XIN(4) Raw X,Y,Z and temperature coordinates that are input to subroutine on this call. On return XIN equals the raw data at time TOUT.
XOUT(4) Smoothed X,Y,Z and optionally temperature calculated at time TOUT.
VOUT(4) First derivative of position and optionally temperature calculated at time TOUT.
AOUT(4) Second derivative of position and optionally temperature calculated at time TOUT.

ISWITCH Switch Values:
0 = IDRSIT switch will not be changed by DRIVER.
1 = IDRSIT Switch will be set equal to 3 by DRIVER so that proper logic path will be taken.

INTERNAL VARIABLES

ICNTF1 Index into raw data arrays X,Y,Z indicating where the raw data is to be stored
ICNTF2 Index into the midpoint arrays (XMID, YMID, ZMID) and velocity arrays (XVM, YVM, ZVM) where new data is to be stored
LTHTSK Total number of frequencies to store for synchronization with POSFIT output
NSKIP Number of raw data points which will be passed to TMPFIT in order to assure that the smoothed data are calculated for the same time point.

CALLING ROUTINE - DRIVER

SUBROUTINES CALLED - POSFIT, TMPFIT
CHANGES

Identical to METROC-K version

COMPILED 8/22/83

DIMENSION XIN(4),XOUT(4),VOUT(4),AOUT(4)
GO TO (10,100,200,200) IDRSIT

INITIALIZE

10 ICNTT=0
ICNTPI=0
ICNTP2=1
CALL POSFIT(IDRSIT,TIN,XIN,TOUT,XOUT,VOUT,AOUT,ICNTPI,ICNTP2,
LTHP1,LTHP2,ISLIDE,ISWTCH)
IF(NDRST.EQ.4) CALL TMFFIT(IDRSIT,TIN,XIN(4),TOUT1,XOUT(4),
VOUT(4),AOUT(4),ICNTT,LTHT,ISLICE)
IDRSIT=2
IF(NDRST.EQ.4) GO TO 30

WHEN ICNTP1=LTHP1 THE RAW DATA ARRAYS ARE FULL, THEREFORE THE
MIDPT AND VELOCITY ARRAYS CAN BE CALCULATED,

15 IF ((ICNTPI.EQ.LTHP1) GO TO 18
ICNTPI=ICNTPI+1
GO TO 19

18 ICNTP2=ICNTP2+1
19 CALL POSFIT(IDRSIT,TIN,XIN,TOUT,XOUT,VOUT,AOUT,ICNTP1,
ICNTP2,LTHP1,LTHP2,ISLIDE,ISWTCH)
IF(NDRST.NE.4) GO TO 20
ICNTT=ICNTT+1
CALL TMFFIT(IDRSIT,TIN,XIN(4),TOUT1,XOUT(4),VOUT(4),AOUT(4),ICNTT,
LTHP1,LTHP2,ISLIDE)
20 ISWTCH=1
RETURN

PROGRAM ASSUMES LTHP1 + LTHP2 WILL BE GREATER THAN LTHT, NSKIT =
HOW MANY RAW DATA POINTS WILL NOT BE PASSED TO THE TMFFIT
SUBROUTINES IN ORDER TO ASSURE THAT THE SMOOTHED DATA ARE
CALCULATED FOR THE SAME TIME POINT.

30 NSKIP=(LTHP1+LTHP2-LTHT-1)/2

LTHTSK = TOTAL NUMBER OF FREQUENCIES TO STORE FOR
SYNCHRONIZATION WITH POSFIT OUTPUT.

100 IF (NSKIP.EQ.0) GO TO 15
IF (ICNTPI.EQ.LTHP1) GO TO 120
ICNTPI=ICNTPI+1
GO TO 125

120 ICNTP2=ICNTP2+1
125 CALL POSFIT(IDRSIT,TIN,XIN,TOUT,XOUT,VOUT,AOUT,ICNTP1,ICNTP2,
LTHP1,LTHP2,ISLIDE,ISWTCH)
NSKIP=NSKIP-1
IF(NSKIP.NE.0) RETURN
GO TO 20

200 IF (ICNTP1.EQ.LTHP1) GO TO 220
ICNTP1 = ICNTP1 + 1
GO TO 225

220 IF (ICNTP2 .EQ. LTHP2) GO TO 225
ICNTP2 = ICNTP2 + 1
C
C IF ALL ARRAYS NEEDED ARE FULL THE SMOOTHED POINT WILL BE
C CALCULATED.
C
225 IF (ICNTP1 .EQ. LTHP1 .AND. ICNTP2 .EQ. LTHP2) IDRSIT = 4
CALL POSFIT(IDRSIT, TIN, XIN, TOUT, XOUT, VOUT, AOUT, ICNTP1, ICNTP2,
& LTHP1, LTHP2, ISLIDE, ISWTCH)
IF (NDRST .NE. 4) GO TO 250
IF (ISLIDE .GT. 1 .AND. ICNTP1 .LT. LTHP1) GO TO 250
IF (ICNTP1 .EQ. LTHP1) GO TO 240
ICNTP1 = ICNTP1 + 1
240 CALL TMPFIT(IDRSIT, TIN, XIN(4), TOUT1, XOUT(4), VOUT(4), AOUT(4), ICNTP1,
& LTHP1, ISLIDE)
250 CONTINUE
RETURN
END
SUBROUTINE DRIVER

This subroutine takes as input the queue of edited data points created by EDIT. For each point, it calculates the uncorrected sensor temperature. Then smoothes the temperature data along with the 3 coordinates of the sonde (X, Y, Z). Calculates the first two derivatives for all four variables. Calculates the winds and relative air speeds. Calculates the corrected air temperature. The air temperature at 100 meter levels is saved for the second part of the subroutine. To conserve memory, the air temperature is stored in the queue in place of the edited data points.

In the second part of the subroutine, initial values of atmospheric pressure are found by 'tying' the lower end of the temperature array to radiosonde data. Then the subroutine backsteps through the temperature array, from low altitude to high, calculating atmospheric pressure and density and printing out the meteorological data.

INPUTS
BRES = 30 WORD ARRAY, RADIOSONDE PRESSURES (MILLIBARS)
BTEMP = 30 WORD ARRAY, RADIOSONDE TEMPERATURES (DEG. CELSIUS)
DRTEMP = 100 WORD ARRAY, DIRECT INPUT SENSOR TEMPERATURE
DRTIM = 100 WORD ARRAY, DIRECT INPUT SENSOR TIME
HEADER = 14 WORD LAUNCH OPERATION LABEL
IWIND = INTEGER SWITCH, 1 = RADAR DATA ONLY
LALT = 30 WORD ARRAY, RADIOSONDE ALTITUDE IN DECAMETERS
NOTEMP = SWITCH, 1 = NO DIRECT TEMPERATURES, -1 TEMP ON CARDS
NPBL = NUMBER OF RADIOSONDE DATA
NPT = TOTAL NUMBER OF DIRECT TEMPERATURE POINTS
TAFOG = TIME IN SECONDS AFTER LAUNCH OF MAXIMUM SONDE ALTITUDE
TIME = TIME IN SECONDS AFTER LAUNCH OF PRESENT DATA POINT
ZAFOG = MAXIMUM SONDE ALTITUDE ENCOUNTERED IN RADAR DATA
ZDD = UPWARD ACCELERATION COMPONENT OF SENSOR
ZS = SMOOTHED ALTITUDE OF SENSOR (M)

OUTPUTS
HEADER, LALT, BTEMP = PRINTED VALUE SAME AS INPUT VALUE
ACC = 800 WORD ARRAY, VERTICAL ACCELERATION AT 100M LEVELS
AIRSPD = VENTILATION VELOCITY OF SONDE (M/S) AIRSPEED
ALTD = ALTITUDE AT 100 METER INTERVALS, 800 WORD ARRAY
ALTG = COMPUTED BASELINE TIE-ON ALTITUDE
BETA = EXPONENT IN LINEAR TEMPERATURE MODEL
BPTINT = ROCKETSONDE TEMPERATURE AT ZLOW
CAEB = TEMPERATURE CORRECTION FOR RADIATION ABSORPTION (DEG C)
CAERO = TEMPERATURE CORRECTION FOR AERODYNAMIC HEATING
CDYN = TEMPERATURE CORRECTION FOR DYNAMIC LAG DEGREE CELSIUS
CEMIS = TEMPERATURE CORRECTION FOR RADIATION EMISSION (DEG C)
CK1, CK2, CK3, CK4 = KRUMINS COEFFICIENTS FOR TEMP. CORRECTION
CTOTAL = TOTAL TEMPERATURE CORRECTION
DAYNGT = FLAG INDICATES DAY OR NIGHT FOR SENSOR (DAY=1.,NIGHT=-1.)
DENS = AIR DENSITY COMPUTED AT PRESENT 100M LEVEL
DRCNM = WIND DIRECTION IN DEGREE AZIMUTH
DTF = TEMPERATURE DIFFERENCE BETWEEN PRESENT AND LOWEST TEMP
EWW = 800 WORD ARRAY, EAST-WEST WIND AT 100M LEVEL (M/S)
FU = 800 WORD ARRAY, FALL VELOCITY AT 100M LEVEL (M/S)
G = GRAVITY COMPUTED BY FUNCTION GRVITY
GAMMA = LAPSE TIME (DEG CELSIUS/METER)
IL = INDEX OF TEMPERATURES
K = INDEX OF RAWINSONDE DATA
LEV = INDEX VALUE OF BASELINE TIE-ON LEVEL IN RADIOSONDE
MIN = TIME IN WHOLE MINUTES AFTER LAUNCH AT PRESENT ALTITUDE
SNW = 800 WORD ARRAY, NORTHWARD WIND COMPONENT (100M LEVELS)
PRES = ATMOSPHERIC PRESSURE (MB) COMPUTED AT PRESENT 100M
RSZM = 30 WORD ARRAY, RADIOSONDE GEOMETRIC ALTITUDE
ISEC = SECONDS CORRESPONDING TO MIN
TAIR = CORRECTED THERMOMETER AIR TEMPERATURE (DEG. CELSIUS)
TCOR = 800 WORD ARRAY, TOTAL THERMOMETRIC CORRECTION 100M/S
TI = CORRECTED MEASURED AIR TEMPERATURE (DEGREES KELVIN)
TMFT = 800 WORD ARRAY, CORRECTED MEASURED AIR TEMPERATURE (CELSIUS)
TMPTRS = 1150 WORD ARRAY, AIR TEMPERATURE (DEG. K) STORED IN .5
SECOND LIST FOR BACK-PROCESSING OF DENSITY AND PRESSURE
WINDE = SMOOTH-DIFFERENTIATED RADAR EASTWARD WIND COMPONENT
WINDN = SMOOTH-DIFFERENTIATED RADAR NORTHWARD WIND COMPONENT
WINT = WIND MAGNITUDE (M/S)
WTAU = TIME CONSTANT OF WIND SENSING SYSTEM (RARACHUTE)
XD = SMOOTHED EASTWARD VELOCITY COMPONENT
XOLD = 800 WORD ARRAY, UNCORRECTED EASTWARD WIND 100M LEVELS
YD = NORTHWARD VELOCITY COMPONENT OF SENSOR (M/S)
YOLD = 800 WORD ARRAY, UNCORRECTED NORTHWARD WIND 100 M/S
ZD = UPWARD VELOCITY COMPONENT OF SENSOR (M/S)
ZOLD = 800 WORD ARRAY, UNCORRECTED UPWARD SENSOR VELOCITY M/S
ZSKM = ALTITUDE (KM)

INTERNAL VARIABLES
BI = OCTAL, BLANKS
DENSS = 800 WORD ARRAY, COMPUTED AIR DENSITIES AT 100M LEVELS
LTHP1, LTHP2, LTH = SMOOTHING INTERVAL LENGTHS
MXTMP = MAXIMUM NUMBER OF ALTITUDE POINTS TO BE STORED AND BACK-PROCESSED
PRESS = 800 WORD ARRAY, COMPUTED ATMOSPHERIC PRESSURE 100M/S
R = GAS CONSTANT
'Recinf' is the input record for one time point
RTOD = DEGREES PER Radian
SCOPY = 13 WORD ARRAY, HOLDS PREVIOUS VALUE FOR DATA POINT
WHEN COMPUTING TEMPERATURE AND X,Y,Z 1ST AND 2ND
DERIVATIVES.
TEMP = SMOOTHED SENSOR TEMPERATURE INTERPOLATED TO 100M LEVEL
WM = AVERAGE MOLECULAR WEIGHT OF AIR
XDD = EASTWARD COMPONENT OF SENSOR ACCELERATION
XS = SMOOTHED EAST COORDINATE OF SENSOR POSITION (M)
ZDECRE = INTERVAL BETWEEN STORED VALUES OF PROCESSED DATA

CALLING ROUTINE - MAIN

SUBROUTINES CALLED
GETEMP, FRQTMP, DIRSIT, WIND, KMNTMP, KMNTAB, GPTOM, GRVITY

COMMENTS = REFERENCE ?METROC-K ALGORITHMS? FOR MATHEMATICAL FORMULAE

NONSTANDARD RETURN TO DRIVER WHEN
MID-INDEX SEARCH VALUE EQUALS UPPER OR LOWER BOUND IN FRQTMP
READ DIRECT TEMPERATURES

The returned values are:

NOTEMP = 1 if no direct temp are given
-1 if direct temp are given
NTPT = the total number of direct temp points given
DRTEMP = array containing the direct temp values
DRTIM = array containing the corresponding time values

CALL GETTEMP1
IDRSIT=1
LINES=50
INDEX=0
ISNATCH=0
IHEAD=1
TAILO=1
OLDATA=FALSE
WRITE (IO,2400) HEADER
WRITE (IO,2420) HEADER

C
CSTART OF EDITED DATA LOOP: GET PT FROM QUEUE
C

20 CONTINUE
CALL REMUV1(F',EMTY) ;get next data pt from queue1
IF (EMTY) GO TO 160 ;no more pts? goto radiosonde tie-on
T = P(1)
X(1) = P(3)
X(2) = P(4)
X(3) = P(5)
X(4) = P(2)
IF (T.LT.TAPOG) GO TO 20 ;if before apogee, get new pt

C
CCONVERT FREQ TO TEMPERATURE
C

IF (NOTEMP) THEN ;if no direct temps and
IF (.NOT.IWIND) ;not a 'winds-only' run
& CALL FRQTMP ( ;change freq to sensor temp X(4)
T, ;time since launch
& X(4), ;frequency input
& X(4), ;sensor temp returned
& FAIL) ;search-fail flag
IF (FAIL) GO TO 20 ;freq outside table? get another pt
ELSE ;if direct temps exist, then
IF (T.LT.DRTIM(1)) GO TO 20 ;if 1st temp not passed get pt
DO 35 I=2,NTF'1 ;search time array for data bracketing T
IF (T.LE.DRTIM(I)) GO TO 40
35 CONTINUE
40 CONTINUE
X(4)=DRTEMP(I)+ (DRTEMP(I-1)-DRTEMP(I)) ;interpolate temp to
& X(T-DRTIM(I))/(DRTIM(I-I)-DRTIM(I)) ;current time
IWIND = FALSE ;be sure wind-only flag not set
END IF

C
CSMOOTH DATA AND FIND DERIVATIVES
C

IF (ISWTCH.EQ.1) IDRSIT=3
CALL DIRSIT (IDRSIT,NDRST,LTHP1, ;smooth data and find derivatives
& LTHP2,LTHT,T,X,TIME,XOUT,YOUT,AOUT,ISLIDE,ISWTCH)
IF (IDRSIT.NE .4) GO TO 20 ;idrsit=4 means enough pts for calc.
XS = XOUT(1) ;put smoothed data into
YS = XOUT(2)
ZS = XOUT(3)
TEMP = XOUT(4)
XD = YOUT(1)
YD = YOUT(2)
ZD = YOUT(3)
TEMFD = YOUT(4)
XDD = AOUT(1)
YDD = AOUT(2)
ZDD = AOUT(3)
TEMFD = AOUT(4)
MIN=TIME/R60P ;minutes since launch for print
ISEC=TIME-MINT60  ;seconds since launch for print
ZSKM=Z/R1E3  ;altitude in km for print
IF (LOPT.GE.2) THEN  ;is detailed printout wanted?
  CALL WIND (WTAU,WINDN1,WINDE1, ;calc winds for print
  AIRSP1,WINDT,DRCTN)
  WRITE (I0,2410) ZSKM,MIN,ISEC, ;and list winds at every pt
  & WINDE1,WINDN1,WINDT,DRCTN,
  & XD,YD,ZD,WTAU
END IF
IF (ZS.LE.Z100M .AND. ;if sonde just crossed 100m level
  OLDATA) THEN  ;and old data exists for interpolation
  CALL WIND (WTAU,WINDN1,WINDE1, ;calc winds and airspeed
  & AIRSP1,WINDT,DRCTN)
  IF (.NOT.IWIND)  ;if not a 'winds-only' run
    CALL KMNTMP (TAIR1,CAERO1,CDYN1, ;find air temperature
    & CEMIS1,CAER01,AIRSP1)
  END IF
C CALCULATE WINDS AND TEMPERATURE AT POINT JUST BELOW 100M LEVEL
  CALL WIND (WTAU,WINDN0,WINDE0, ;find wind components & airspeed
  & AIRSP0,WINDT,DRCTN)
  IF (.NOT.IWIND)  ;if not a 'winds-only' run
    CALL KMNTMP (TAIRO,CAERO0,CDYN0, ;find air temperature
    & CEMISO,CASE00,AIRSP0)
  END IF
C CALCULATE WINDS AND TEMPERATURE AT POINT JUST ABOVE 100M LEVEL
  CALL SWAP (SCOPY)  ;swap old data into XS,YS,...TEMP0
  CALL WIND (WTAU,WINDN0,WINDE0, ;find wind components & airspeed
  & AIRSP0,WINDT,DRCTN)
  IF (.NOT.IWIND)  ;if not a 'winds-only' run
    CALL KMNTMP (TAIRO,CAERO0,CDYN0, ;find air temperature
    & CEMISO,CASE00,AIRSP0)
  END IF
C INTERPOLATE DATA TO 100M LEVEL
50 CONTINUE  ;begin interpolation loop
  RATIO = (Z100M-SCOPY(4))/ZS-SCOPY(4))
  T100Z = SCOPY(1) + RATIO*TIME-SCOPY(1))
  XOLD = SCOPY(6) + RATIO*XD-SCOPY(6)
  YOLD = SCOPY(7) + RATIO*YD-SCOPY(7)
  ZOLD = SCOPY(8) + RATIO*ZD-SCOPY(8)
  FV = -ZOLD
  EWW = WINDE0 + RATIO*WINDE1-WINDE0
  SNW = WINDN0 + RATIO*WINDN1-WINDN0
  ACC = SCOPY(12) + RATIO*ZDD-SCOPY(12)
  CALL KMNTAB (Z100M,CK1,CK2,CK3,CK4,T100Z,FAIL)
  IF (FAIL) THEN
    CK1 = R9999
    CK2 = R99999
    CK3 = R99999
    CK4 = R99999
  ELSE
    CK1 = CK1 * R1E4
    CK3 = CK3 * R1E11
  END IF
  IF (.NOT.IWIND) THEN
    WRITE (I0,2010) ZSKM,C1'%1,CK2,CK3,CK4
    TSEN = SCOPY(5) + RATIO*TEMP-SCOPY(5))
    TCO = TAIR - TSEN
    CAERO = CAERO0 + RATIO*(CAER01-CAERO0)
  ELSE
    WRITE (I0,2010) ZSKM,C1'%1,CK2,CK3,CK4
    TSEN = SCOPY(5) + RATIO*TEMP-SCOPY(5))
    TCO = TAIR - TSEN
    CAERO = CAERO0 + RATIO*(CAER01-CAERO0)
CDYN = CDYNO + RATIO*(CDYN1-CDYNO)
CEMIS = CEMISO + RATIO*(CEMISI-CEMISO)
CASB = CASB0 + RATIO*(CASB1-CASB0)
TAIRC = TAIR - R273P1 ; air temp in celsius
TSENC = TSEN - R273P1 ; sensor temp in celsius
WRITE (IO,2430) ZSKM,TAIR,TSEN, ; temp data printout
&
TCOR,CAERO,CDYN,CEMIS,CASB,
&
CK1,CK2,CK3,CK4,DAYNGT,TAIRC,TSENC
END IF

C--------------------------------------
C SAVE AIR TEMPERATURE FOR PRESSURE AND DENSITY CALCULATION
C--------------------------------------
M(1)=EWW
M(2)=SNW
M(3)=Z100M
M(4)=TAIR
M(5)=TSEN
CALL ADD2 (M,FULL)
IF (FULL) THEN
   PRINT 2000, Z100M
5000 FORMAT ('*x* QUEUE OVERFLOW IN DRIVER',
&        'AT ALTITUDE Z100M=',F6.0,' METERS')
STOP
END IF
ZLOW = Z100M ; save lowest level reached
Z100M = Z100M + ZDECRE ; decrement 100m level
IF (ZS.LE.Z100M) GO TO 50 ; new level bracketed? interpolate again
END IF
CALL SWAP (SCOPY) ; save old smoothed data in SCOPY
OLDATA = TRUE ; set 'old data available' flag
GO TO 20 ; get another edited data point
C------------------------
C END OF EDITED DATA PROCESSING LOOP
C------------------------

C RADIOSONDE TIE-ON
C------------------------
106 CONTINUE
160 IK=INDX
WRITE (IO,2432)
2432 FORMAT(/10X,'**** NO TEMPERATURE CORRECTION A',
&            'PLIED OUTSIDE THE KRUMINS TABL',
&            'E RANGE (20-70 KM) *****/**/*)
INDX=INDX-1
IL=INDEX
IF (IWIND) GO TO 1166
DO 161 I = 1,NPBL
ISTORE = LALT(I) x R10P
ISTORE = GPTOM (ISTORE, XLAT) + ROP5
RSZM(I) = ISTORE
161 CONTINUE
ALTG=RZER0
C *SCAN DATA FOR ERRORS, LINEAR INTERPOLATION OF TEMPERATURE GAPS
C+ LESS THAN 3KM
C+ CHECK 3-KM GAP BETWEEN R/S AND ROCKETSOSNDE ....
K=1
IF((ZLOW-RSZM(1)).GT.R3E3) GO TO 1166
IF(ZLOW.GT.(RSZM(1)+R1E2)) GO TO 1144
WRITE (IO,2160)
2160 FORMAT (IA1)
WRITE (IO,2161)
2161 FORMAT (///15X,'RAWINSONDE',30X,'ROCKETSONDE',22X,'TEMP', DIFF',2X, & & 4X,'INTERPOLATED'/2X, & & 'LEVEL METERS',7X,'G',8X,'BTMP',10X,'PT',5X,'ALT',6X,'TMPTRS', & & 6X,'TMPT',12X,'DTP',9X,'BTMP AT ALT',')
K=NPBL
C... LOOK FOR TWO CONSECUTIVE R/S LEVELS WHICH BRACKET ZLOW....
1140 IF(ZLOW.LT.RSZM(K)) GO TO 1141
K=K-1
1141 KK=K
K=K+1
IF(KK.LE.0) KK=K+1
C... CHECK 2.5 DEGREE DIFF, BETWEEN ROCKETS ONE TEMP, AT ZLOS AND & THE CORRESPONDING R/S TEMP, IN THE R/S BRACKET....
1142 BTPINT=BTMP(KK)+(BTMP(KK)-BTMP(K))*ZLOW-RSZM(KK)/(RSZM(1)- & RSZM(KK))
STORE = BTPINT - DFILE(1,IL) + R273P1
DTP = ABS(STORE)
ALTD = ALTI(ZHIGH,ZDECRE,IL)
STORE = LALT(K)*R10P
TAIRC = DFILE(1,IL)-R273P1
WRITE (IO,2162)K,RSZM(K),STORE,BTMP(K),IL,ALTD, & TAIRC,TAIRC,DTP,BTPINT
2162 FORMAT(3X,2(I4,3F10.2,8X),FB.2,10X,F10.2)
1143 LEV=K
ALTG=RSZM(K)
GO TO 1165
C... PREFORM LINEAR INTERPOLATION ON TEMP, TOFILL THE TEMP GAP WHERE & THE ALT. GAP BETWEEN R/S AND ROCKETSONDE IS LESS THAN 3KM ....
1144 WRITE(IO,2163)
2163 FORMAT(/10X,'*** THE FOLLOWING INFORMATION ARE RELATIVE', & 'D TO INTERPOLATION DUE TO N', & 'OVERLAP WITH R/S ***')
LEV=1
ALTG=RSZM(1)
DTMP=(DFILE(1,INDX)-BTMP(1)-R273P1)/((ZLOW-RSZM(1))/R1E2)
1155 INDX=INDX+1
IL=INDX
ZLOW=ZLOW+ZDECRE
ISTORE = INDX - 1
DFILE(1,INDX)=DFILE(1,ISTORE)-DTMP
IF(ZLOW.GE.(RSZM(1)+R1E2)) GO TO 1155
K=0
GO TO 1141
1165 IF(LEV.LE.0) GO TO 1166
IF(ALTG.GT.RZERO) GO TO 167
1166 LEV=0
IL=1
WRITE(IO,2168)
2168 FORMAT(/10X,80(' '),10X,'*** NO PRESSURE AND DENSITY COMPUTATION DUE TO FAILURE ON 2.5 DEG', & 'TEST OF TEMPERATURE ***/10X, 80(' ')////)
I=K-1
STORE = LALT(I) * R10P
IF(K.GT.1) WRITE( IO,2162) I,RSZM(I),STORE,BTMF(I)

C+ *COMPUTE GRAVITY, PRESSURE AND DENSITY

PRS=BI
DNS=BI
ZLOW=ALTI(ZHIGH,ZDECRE,IL)
DO 168 I=IL,INDX
II=INDX-I+IL
PRESS(II)=PRS
DENSS(II)=DNS
168 CONTINUE

LN=50
IF(IL.GT.1) GO TO 1167
IXS=INDX
GO TO 206
1167 CONTINUE

GAMMA = RZERO
G = GRVITY(ZLOW)
BETA = RZERO
TAIR = DFILE(1,IL)
E:TMPK=BTMP(K)+RZ73P1
F'RES =
  BF'RES(LEV)*((TAIR/TAIR1)**W*GRVITY(ZLOW)*(ZLOW-RSZM(LEV)))/(R*(TAIR-BTMPK))
IXS=INDX
NKPT = 1
I = IL
GO TO 180

C BACK STEP THRU THE ARRAY OF TEMPERATURES

170 CONTINUE
I = I - 1
PRS=RZERO
DNS=RZERO
IF (I .EQ. 0) GO TO 200
TAIR = DFILE(1,I)
ZLOW = ZLOW - ZDECRE
GAMMA = (TAIR - TAIR1) / (-ZDECRE)
G = GRVITY(ZLOW)
BETA = -W*G / (R*GAMMA)
PRES = PRES1 * (TAIR/TAIR1) ** BETA
180 DENS = PRES*WM / (R*TAIR)
DENS=DENS*R1E5
ZSKM = ZLOW / R1E3

C ALL FACTORS ARE READY FOR PRINT

LINES = 0
WRITE (IO,2450) HEADER
TAIR1 = TAIR
PRES1 = PRES
IPD=1
PRS=PRES
DNS=DENS
ALTD = ALTI(ZHIGH,ZDECRE,I)
IF (ALTD .LT. R7E401) GO TO 193
DNS=R999P9
PRS=R999P9
193 CONTINUE
PRESS(I) = PRS
DENSS(I) = DNS
IJ=IJ-1

NORMAL TERMINATION

200 IF(IPD.EQ.0) GO TO 202
GO TO 206

202 CONTINUE
ALTD = ALTI(ZHIGH,ZDECRE,I)
IF (ALTD .LT. R7E401) GO TO 203

203 CONTINUE
IF (I .GT. 798) GO TO 204
PRESS(I) =PRS
DENSS(I) =DNS
IK = I

204 CONTINUE
INDX=IK
GO TO 207

206 CONTINUE
IF (IWIND) GO TO 207

207 CONTINUE
IF(I.GT. INDX) GO TO 174

210 WRITE(IO,2463) ALTD, EWH(I),SNW(I),FV(I),DFILE(I,I)-R273P1,PRS,
& DNS,ACC(I),XOLD(I),YOLD(I),ZOLD(I),TCOR(I),GAMMA,BETA,G

2463 FORMAT (1X,F7.0,F8.2,5F9.2,F8.2,4F9.2,E12.3,1X,2F7.2)

CONTINUE LOOPING UNTIL ALL DATA IS PROCESSED.

WHEN IWIND = 1 , THIS IS A WINDS ONLY CASE,
THE PROGRAM MUST NOT BE ALLOWED TO BACKSTEP THRU THE ARRAY OF
TEMPERATURES. STATEMENT 170

INSTEAD REDUCE THE COUNT BY 1 AND GO TO STATEMENT 207.

174 CONTINUE
IF ((.NOT. IWIND) .AND. (I .GT. 1 )) GO TO 170
IF (IWIND) .AND. (I .GT. 0 )) I = I-1
IF (IWIND) .AND. (I .GT. 0 )) GO TO 207

RETURN
C

2400 FORMAT ('1FLIGHT IDENTIFIER# ', 14A6 //
& 'ALT', 4X, 'TIME', 10X, 'WIND (MET/SEC, DEG)',
& 9X, 'VELOCITY', 6X, 'AIRSPEED RESPONSE', /
& (KM), (MIN/SEC), EAST NORTH TOTAL DRCTN',
& 4X, 'EAST NORTH', 4X, 'UP', 13X, '(SEC)'/)

2410 FORMAT (1X, F5.2, 1X, I2, '-', F5.2, 3(1X, F6.1),
& ' ', F5.1, 1X, 4(1X, F6.1), 4X, F5.1, 6X, I3, F9.3)

2420 FORMAT ('1FLIGHT IDENTIFIER# ', 14A6 //
& 'ALT', TEMPERATURE (K), 11X 'CORRECTIONS',
& 15X, 'KRUMINS COEFFICIENTS', 14X, 'TMPRTRE (CELSIUS)',
& '(KM)', 4X, 'AIR SENSOR ',
& 'TOTAL AERO, DYN, RAD (OUT/IN)', 3X,
& 'K1*E4, K2', 4X, 'K3*E11 K4 DAY/NIGHT AIR SENSOR'/)

2430 FORMAT (1X, F4.1, 1X, 2(2X, F5.1), 2X, 5(1X, F5.1),

2010 FORMAT (' ', F4.1, 50X, F4.2, 2X, F5.3, 2X, F6.2, 2X, F5.2)

END
SUBROUTINE EDIT

===============================================

PURPOSE

This subroutine retrieves the .1 second raw data from the
input queue as it becomes available, averages sets of NAVG
raw points into clean edited data points, and adds them
to the edited data queue.

CONSTANTS

DINERT

"inert" value. A large constant, alternatingly
positive and negative, which is substituted for
the frequency datum of noisy or reference
points. The filter DIRSIT will "coast"
(quadratically extrapolate) through these
unaffected by their amplitude.

INOISE

=1. Value of data-classification flag indicating
noise

IREF

=2. Value of data-classification flag indicating
reference signal

ISIG

=3. Value of data-classification flag indicating
thermistor signal

IUHE

Bottom index of upper half of DSTOR

IUHT

Top index of upper half of DSTOR

STOP1

Stop time plus 12; METROC requires a 12 second
buffer at the end of the data to be processed

INTERNAL VARIABLES

BLWR

Lower bound of tracking gate (Hertz)

BND

Upper bound of each overlapping 10Hz band
used in searching for signal

BUPR

Upper bound of tracking gate (Hertz)

D

Sonde frequency used to compare against bounds
of tracking gate (temporary storage)

DIFERT

Difference between the absolute values of
a point's frequency value and the "inert"
value. Used to detect if point has been
given "inert" value.

DSTOR(5,10)

A 10 point sample of raw .1 second input data

DSTOR(1,j) = GMT time of data pt (seconds)

DSTOR(2,j) = frequency datum (Hertz)

DSTOR(3,j) = azimuth (degrees)

DSTOR(4,j) = elevation (degrees)

DSTOR(5,j) = range (meters)

DSUBST

Mean value of raw data points which lie
in the tracking gate. Used as a substitute
for points which lie outside the gate, and
to adjust SIGLEV.

DTRF

Time at midpoint of reference interval. Used to
calculate sums for least squares fit (same as TRF)

EDATA(5)

A clean edited data point ready to be stored
in the edited data queue

EDATA(1) = time since launch (seconds)

EDATA(2) = frequency datum (Hertz)

EDATA(3) = X coordinate of sonde (meters)

EDATA(4) = Y coordinate of sonde (meters)

EDATA(5) = z coordinate of sonde (meters)

EOD

End-of-data flag returned by ADVANC. Values:
1= flight is over, no more raw data coming
-1 = more data coming

FULL
Full queue flag returned by ADD1. Values:
1 = edited data queue is full; current pt not stored
-1 = queue not full

I
General DO loop index

IB
Index to bottom half of DSTOR

IBND
Index of most populous 10Hz band in searching for signal.

IFLAG
Data-classification flag. Indicates whether edited data point is noise (=1), reference (=2), or thermistor signal (=3). Used for printout.

INCATE(10)
In MET signal tracking, the flag indicating whether each .1 second data point in DSTOR(3,j) is in (=1) or not in (=0) the signal tracking gate.

INTRVL
Number of points in the queue to be replaced with "inert" value, starting with last point put in.

ISKREF
Used to compute initial points of signal to be skipped when replacing reference data with "inert" value.

ISS
Signal point counter. Number of edited data points accepted as thermistor signal.

IT
Index to top half of DSTOR

J
General DO loop index

JR
Reference point counter. Counts number of reference points since last signal interval. JR=0 during signal interval.

JRP
Reference interval counter. Counts number of reference intervals processed (at least 5 consecutive reference points constitute a reference interval)

KB
Number of points in DSTOR with frequency less than upper bound BND of 10 Hz band.

KBL
Number of points in DSTOR with frequency less than BND - 5 Hz

KBLL
Number of points in DSTOR with frequency less than BND - 10 Hz (not initialized until BND = 25)

KEND(32)
In searching for MET signal, population in 29 overlapping 10 Hz bands: KEND(3)= 20-30 Hz, KEND(4)= 25-35 Hz, ..., KEND(31)= 160-170 Hz

LOS
In tracking signal, number of edited noise points since last signal or reference point. (not used)

NAFTR
Verify interval counter. Counts signal points after a reference interval, verifying termination of the reference interval.

NGATE
1 plus the number of raw data points in DSTOR(2,j) whose frequencies lie within the signal tracking gate (SIGLEV+HGATE)

REF
Mean frequency of a reference interval (Hertz), excluding the first and last points. Used to compute sums for least squares fit to ref function.

RS
Frequency value of the last reference point in a reference interval

RSUM
Sum of frequency values in a reference interval, excluding the first point in the interval. Used
for computing mean value over the interval.

RT1 Time after launch of the second point in a reference
interval (seconds)

RT2 Time after launch of the next to last point in
a reference interval (seconds)

RT3 Time after launch of the most recent reference
point received. Saved for RT2 if next point
is also reference, discarded if last point in
reference interval.

SUM(5) Temporary sums for averaging raw .1 second data
in bottom half of DSTOR to get edited point (EDATA)

    SUM(1) (not used)
    SUM(2)= sum of frequencies in DSTOR(2,j)
    SUM(3)= sum of azimuths in DSTOR(3,j)
    SUM(4)= sum of elevations in DSTOR(4,j)
    SUM(5)= sum of ranges in DSTOR(5,j)

SUMGTE Accumulator for computing mean of frequencies
within the gate when tracking signal.

TDSTRT Elapsed time since launch of raw data (seconds)

TEST Average of frequencies in bottom half of DSTOR
Used to tell if 170<=TEST<=200.

TLAPS Elapsed time since TSTART of raw data (seconds)
Used to detect starting point.

TRF Time at midpoint of reference interval, calculated
between second and next to last edited pt in intrvl

COMMON VARIABLES

B(3) Accumulated parameters used by POLYFT to
compute quadratic least squares fit to
reference data.

B(1)= sum of (ref interval frequency)
B(2)= sum of (ref interval frequency x
ref interval midpoint time)
B(3)= sum of (ref interval frequency x
(ref interval midpoint time) xx2)

BT(5) Accumulated parameters used by POLYFT to
compute quadratic least squares fit to
reference data.

BT(1)= number of reference intervals
BT(2)= sum of (ref interval midpoint time)
BT(3)= sum of (ref interval midpoint time xx2)
BT(4)= sum of (ref interval midpoint time xx3)
BT(5)= sum of (ref interval midpoint time xx4)

HGATE Half-width of signal gate used for tracking
thermistor signal (Hertz)

IO1 Output device number for informative messages.
Possible values:
  99= dummy device. No output occurs when used
  101= terminal display
  102= printer

IWIND Winds-only flag. Possible values:
  1= radar data only will be processed
  -1= temperature data will also be processed

NAVG Number of raw data points averaged to obtain an
edited data point (cannot exceed 1/2 the dimension
of DSTOR) Was 5 in original program.

SIGLEV Center frequency of signal gate used for tracking
thermistor signal (Hertz). Starts at approximate
value (155) and is adjusted to follow signal.

TAPOG
Time in seconds after launch of maximum sonde altitude encountered

TSTART
Time in seconds after launch to start processing data (selected to occur after payload ejection)

TSTOP
Time in seconds after launch to stop processing data

APOG
GMT launch time (seconds)

Maximum sonde altitude (apogee) encountered (meters)

SUBROUTINES CALLED
ADVANC, TIMCHK, REWRT, RDRCRD, ADD1, POLYFT

CALLED BY - MAIN

TERMINATION CONDITIONS
If number of reference intervals is less than 3/ prints 'Error in Edit - Less than 3 reference values available'

If terminal 'S' key is pushed in ADVANC/ prints 'End of real time processing' and returns

If edited data queue is full/ prints 'Queue is full'

'End of real time processing' and returns

CHANGES
This is an extensively modified version of the METROC-K subroutine EDIT. The modifications made are:
1. The raw data is input from the raw data queue via subroutine ADVANC, instead of from tape via subroutine INTAPE as in the old routine.
2. All parts dealing with FSK time have been deleted.
3. The "replacement A, E, R data" sections have been deleted.
4. The number of raw data points in the average, previously 5, has been made adjustable with the variable NAVG. This allows run-time experimentation to determine the proper data rate to avoid exhausting queue capacity before the end of the flight.
5. Edited data points are stored in the edited data queue by subroutine ADD1, instead of being stored on tape as in old routine. Array DFILE, which was used to accumulate data for writing to tape, has been eliminated.
6. New subroutine REWRT accesses the queue to replace reference dwell frequencies with "inert" values. In the old routine this was done in array DFILE.
7. The order of statements in the subroutine has been changed around to "untangle" it and eliminate unnecessary GO TO statements.

COMPiled 8/16/83

REFERENCES
INCLUDE 'RCOMMON.SA1'
DIMENSION DSTOR(5,10) ;second dimension must be at least 2*NAVG
DIMENSION INGATE(10) ;dimension must be at least 2*NAVG
DIMENSION EDATA(5), KEND(32), SUM(5)
C

INITIALIZE PARAMETERS

C-------------------------------------

DATA NAFTRO/ 5/
DATA ISIG/ 1/, IREF/ 2/, INOISE/ 3/

IUHE=NAVG+1 ;lowest index in upper half of DSTOR
IUHT=2*NAVG ;top index in upper half of DSTOR
TSTOP=TSTOP+12. ;program needs 12s buffer at end of data
JK=0 ;edited data point counter
JR=0 ;reference point counter
JRP=0 ;reference interval counter
IFRCK=0 ;noise point counter
ISKREF=0
NAFTR=0 ;verify interval counter
RSUM=0. ;ref frequency accumulator
ISS=0 ;signal point counter
DINERT=999.99 ;"inert" value
KBL=0
KL=0
LOS=0
ZAPOG=0. ;apogee altitude

DO 30 I=1,3 ;sums and coefficients
  C(J)=0.
  B(J)=0.
  ST(J)=0.
30 CONTINUE

BT(4)=0.
BT(5)=0.

C--------------------------------------------------
C READ TO START OF DATA
C AND FILL UPPER HALF OF DSTOR ARRAY
C--------------------------------------------------

DO 50 I=IUHE,IUHT
  CALL ADVANC ( DSTOR(1,I), time
  & DSTOR(2,I), frequency
  & DSTOR(3,I), azimuth
  & DSTOR(4,I), elevation
  & DSTOR(5,I), range
  & EOD ) ;end-of-data flag
  IF (EOD) GO TO 200 ;no more data? end real time proc.
  TLAPS=DSTOR(1,I)-TZERO
  TDIFF=TDIFF-TSTART ;elapsed time since TSTART
  IF (TDIFF.LT.0.25) GO TO 40 ;not past start time? start over
50 CONTINUE

The raw data points should come at .1 second intervals. TIMCHK checks the time values and corrects any dropouts. If there are too many dropouts the program is stopped.

CALL TIMCHK (DSTOR,NAVG,4)

C--------------------------------------------------
C BEGINNING OF REAL TIME LOOP
C--------------------------------------------------

C Each iteration the .1 second raw data
points in the top half of DSTOR are shifted
to the bottom half for processing, and a
new set is put in the vacated top half by
ADVANC.
ADVANC gets a point from the raw data
queue; if there are none it waits for one to
come in. If the terminal's S key has been
pushed, it returns EOD = 1, indicating that
real time processing is finished.
The .1 second raw data points are processed
in sets with NAVG points per set (in previous
version NAVG was 5). Later each set
will be averaged into a single edited data point.

60 CONTINUE ; start of main loop
DO 80 IB=1,NAVG ; get a set of raw data pts
IT=IE+NAVG
DO 70 J=1,5 ; shift point from top half
DSTOR(J,IB)=DSTOR(J,IT) ; to bottom half of DSTOR
70 CONTINUE
CALL ADVANC ( ; get next pt from raw data queue
& DSTOR(1,IT), ; time
& DSTOR(2,IT), ; frequency
& DSTOR(3,IT), ; azimuth
& DSTOR(4,IT), ; elevation
& DSTOR(5,IT), ; range
& EOD) ; end-of-data flag
IF (EOD) GO TO 200 ; no more data? end real time proc.
80 CONTINUE
CALL TIMCHK (DSTOR, IUHT, 5); if more than 5 time errors, abort
IF (IWIND.EQ.1) GO TO 69 ; winds only? skip tracking gate
IF (JR.NE.0) GO TO 69 ; in reference? skip tracking gate

C-----------------------------------------------------------
C TRACKING GATE PROCESSING
C-----------------------------------------------------------

664 CONTINUE
BUFR=SIGLEV+HGATE ; tracking gate upper bound
IF (BUFR.GT.170.) BUFR=170. ; 170 is upper limit of sig.freq
BLWR=SIGLEV-HGATE ; tracking gate lower bound
IF (BLWR.LT.20.) BLWR=20. ; 20 is lower limit of sig.freq
NGATE=1 ; # of pts within gate + 1
SUMGTE=SIGLEV ; SIGLEV is included in sum
DO 671 J=1, IUHT ; check all 10 pts in DSTOR
INGATE(J)=0
D=DSTOR(2,J) ; frequency of point J
IF (D.LE.BUFR .AND. D.GE.BLWR) THEN ; if freq is within gate,
INGATE(J)=1 ; set in-gate flag of pt J
SUMGTE=SUMGTE+D ; add freq of pt J to sum
NGATE=NGATE+1 ; increment pt counter
IF (NGATE.GE.6) GO TO 672 ; only average first 5 pts in gate
END IF
671 CONTINUE
672 CONTINUE ; exit from loop; >5 pts in gate
C-------------------------------
C SIGNAL NOT IN GATE: SEARCH FOR SIGNAL
C-------------------------------
IF (NGATE.LE.2) THEN ; if 2 or less pts in gate, search
END = 20.
IBEND = 1
DO 662 I = 1, 31  
   $search 29 overlapping 10Hz bands
   KB = 0
      DO 661 J = 1, IUHT
         IF (DSTOR(2, J) .LT. END) KB = KB + 1
         KBND(I) = KB - KBLL
         KBL = KBL
         END = END + 5,
         IF (I .GE. 3 .AND.
            & KBND(I) .GT. KBND(IBND)) IBND = I
      CONTINUE
   IF (KBND(IBND) .GE. 3) THEN  
      SIGLEV = (IBND + 2) * 5  
      ;shift gate to center of that band,
      ;and go to tracking gate process.
      GO TO 664
   END IF
   TEST = 0.
   ; pnt is assumed to be ref or noise
   DO 665 K = 1, NAVG
      TEST = TEST + DSTOR(2, K)
   CONTINUE
   TEST = TEST / NAVG
   ;mean freq of bottom half of DSTOR
   IF (TEST .LT. 170. OR.
      & TEST .GT. 200.) THEN  ;it is noise. In this case*
      DO 663 Y = 1, NAVG
         DSTOR(2, K) = DINERT  ;replace with alternating inert values
      CONTINUE
      DINERT = -DINERT
      LOS = LOS + 1  ;increment noise point counter
      END IF
   END IF
C-----------------------------------------------
C SIGNAL IN GATE: ADJUST CENTER FREQUENCY
C-----------------------------------------------
IF (NGATE .GT. 2) THEN  
   DSUBST = SUMGTE / NGATE  ;mean freq of pts in gate
   LOS = 0  ;reset noise point counter
   DO 673 J = 1, IUHT
      IF (INGATE(J) .EQ. 0) DSTOR(2, J) = DSUBST  ;for pts outside gate.
   CONTINUE
   SIGLEV = (SIGLEV + DSUBST) * 0.5  ;adjust gate center freq
END IF
C-----------------------------------------------
C AVERAGE RAW DATA TO OBTAIN EDITED DATA POINT
C-----------------------------------------------
The set of .1 second raw data points in the
bottom half of DSTOR will be averaged to obtain
a single edited data point, EDATA. The number of
raw points averaged is NAVG (was 5 in METROC)
so the sampling period of the edited points is
NAVG * .1 second (was .5 sec in METROC)

69 CONTINUE
   JK = JK + 1
   DO 700 J = 1, 4
      SUM(J) = 0.
   CONTINUE
   DO 710 J = 1, NAVG
      $sum data in bottom half
SUM(2) = SUM(2) + DSTOR(2,J)  ; of DSTOR for average
SUM(3) = SUM(3) + DSTOR(3,J)
SUM(4) = SUM(4) + DSTOR(4,J)
SUM(5) = SUM(5) + DSTOR(5,J)

710 CONTINUE
EDATA(1) = (DSTOR(1,1) - TZERO) ; time (since launch) taken at
+ (NAVG - 1) * 0.05  ; middle of average.
EDATA(2) = SUM(2) / NAVG ; Average by dividing by
EDATA(3) = SUM(3) / NAVG ; number of pts summed
EDATA(4) = SUM(4) / NAVG
EDATA(5) = SUM(5) / NAVG

C-----------------------------------------------------------
C Data reduction method requires a 12 second
C buffer on the end of the data. If time is
C greater than TSTOP1=TSTOP+12, stop real time
C processing
C-----------------------------------------------------------
IF (EDATA(1).GT.TSTOP1)
  WRITE (I01,1040) ; 'stop time reached'
1040 FORMAT ('STOP TIME REACHED')
  IF (EDATA(1).GT.TSTOP1) GO TO 200 ; end real time processing
IF (IWIND.EQ.1) EDATA(2) = 0.0 ; if winds only, zero the freq
IF (IWIND.EQ.1) GO TO 150 ; if winds only, skip classifi-
C ; cation of signal
---------------------------------------------------------------
DIFERT = ABS(EDATA(2)) - ABS(DINERT) ; DIFERT is zero if point
DIFERT = ABS(DIFERT) ; has been given "inert" value
IF ((EDATA(2).GE.10. .AND. ; if freq is in signal range (10-170Hz)
  .AND. ; or was given "inert" value,
  DIFERT.LT.0.1) THEN ; treat it as signal.
    IFLAG = ISIG
    ISS = 1 + ISS ; increment signal point counter
    IF (JR.EQ.0) GO TO 150 ; If last ref intrvl has been
    processed, no processing is needed
    C---------------------------------------------------------------
C PROCESS LAST REFERENCE INTERVAL
C---------------------------------------------------------------
IF (NAFTR.EQ.0) THEN ; if end of ref interval is verified
  NAFTR = NAFTR0 ; reset after-ref pt counter (to 5)
  IF (JR.GE.5) THEN ; dont use ref intrvl of less than 5 pts
    JRP = JRP + 1 ; increment ref interval counter
    TRF = (RT1 + RT2) * .5 ; time at midpt of ref interval
    REF = (RSUM - RS) / (JR - 2); avg of freqs in ref interval,
    C ; (excluding endpts)
    DTRF = TRF
    DREF = REF ; increment sums for quadratic
    BT(1) = BT(1) + 1.0 ; least squares fit
    BT(2) = BT(2) + TRF
    BT(3) = BT(3) + TRF*TRF
    BT(4) = BT(4) + TRF*TRF*TRF
    BT(5) = BT(5) + TRF*TRF*TRF*TRF
    B(3) = B(3) + DREF
    B(2) = B(2) + DREF*DTRF
    B(1) = B(1) + DREF*DTRF*DTRF
C END IF
END IF
NAFTR=NAFTR-1 ;decrement after-ref pt counter
IF(NAFTR.NE.0) GO TO 150 ;if still in verify interval, exit

C
C REPLACE REFERENCE FREQUENCIES WITH 'INERT' VALUE
C
C At this point, 5 (NAFTR0) signal points have been received since the last reference point, so the reference interval is assumed ended. Subroutine RENRT moves back a given number of points in the queue and replaces the frequency datum of those points with the "inert" value DINERT. The number of points rewritten, INTRVL, extends from about the 5th point preceding the reference interval to the 4th point following it. The overlap is to mask any switching transients.

Number of data points to rewrite (INTRVL) =
length of reference interval (JR)
- noise pts since last ref interval(ISKREF)
+ 2 'verify' intervals,
    one on each side of ref interval(2×NAFTR0)
- 3 points.

INTRVL=JR-ISKREF+2×NAFTR0-3 ;nbr of edited pts to rewrite
CALL RENRT (INTRVL,DINERT) ;replace freqs
JREF = 0 ;zero reference point counter
ISKREF=0
RSUM = 0.0 ;zero sum for reference freq avg
GO TO 150 ;go to coordinate conversion sect.

END IF

C
C DATA IS REFERENCE
C
C IF (EDATA(2).GE.170 .AND. ;if freq is within reference &
    EDATA(2).LE.200.) THEN ;range(170-200Hz), it is reference.
    IFLAG = IREF
C
C If ISS=0, no signal data has been found -- process cannot start in reference. It should be noted that this can happen only at the beginning of the first record.

C IF (ISS.EQ.0) GO TO 60 ;if no signal, get more raw data
C
JR = JR + 1 ;increment reference pt counter
IF (JR.GE.2) THEN ;leave 1st pt in interval out of avg
    RSUM = RSUM + EDATA(2) ;sum of ref freqs for avg
    RS = EDATA (2) ;save last ref freq for avg
    IF (JR.EQ.2) RT1=EDATA(1) ;save time of 2nd pt for time calc.
    RT2 = RT3 ;time of next to last pt
    RT3 = EDATA(1) ;time of last pt in ref interval
END IF
GO TO 150

C
C DATA IS NOISE
IF (EDATA(2).LT.10. .OR. EDATA(2).GT.200.) THEN  
  IFLAG = INOISE  
  EDATA(2) = DINERT  
  ISKREF = ISKREF+1  
END IF

IF (EDATA(2).LT.10. .OR. EDATA(2).GT.210.) THEN  
  IFRCK = IFRCK + 1  
END IF

C---------------
C CONVERT TO RECTANGULAR COORDINATES
C------------------•---------------------------------------------
C 'RDRCRD' converts the azimuth, elevation, and range
coordinates in EDATA(3), EDATA(4), EDATA(5) into
rectangular X,Y,Z coordinates, returning the three
values back in EDATA(3), EDATA(4), EDATA(5),
respectively.
C---------------

150 CONTINUE
CALL RDRCRD (EDATA(3),EDATA(4),EDATA(5))

C---------------
C FIND APOGEE
C---------------------------
IF (EDATA(5).LT.ZAPOC) THEN  
  ZAPOG = EDATA(5)  
  TAPOG = EDATA(1)  
END IF

C---------------
C ADD EDITED DATA POINT TO QUEUE
C--------------------
CALL ADD1 (EDATA,FULL)  
IF (FULL.GT.0.) THEN  
  WRITE (IO1,1010)  
  1010 FORMAT ('EDITED DATA QUEUE IS FULL')  
  GO TO 200  
END IF

GO TO 60

C---------------
C END OF REAL TIME PROCESSING
C---------------------------
200 CONTINUE
WRITE (IO1,1000)  
1000 FORMAT ('END OF REAL TIME PROCESSING')

IF (IFRCK.GT.1000 .AND. .NOT.IWIND) THEN  
  WRITE (IO1,1030)  
  1030 FORMAT ('*** MORE THAN 1000 BAD FREQ. DATA POINTS FOUND')  
  IWIND=1  
END IF

IF (IWIND) RETURN  

C---------------
C CALCULATE LEAST SQUARES COEFFICIENTS
C----------------------------------------------------------

IF (JRP.LT.3) THEN ;less than 3 ref intervals?
  WRITE (I01,1020) ;display error message
1020  FORMAT ('xxx ERROR IN EDIT'/
               & 'LESS THAN 3 REFERENCE VALUES ARE AVAILABLE'/
               & 'TO CALCULATE LEAST SQUARES FIT')
  STOP
END IF
CALL POLYFT (3) ;least squares coefficients
END
SUBROUTINE FITON(ICNTPZ,LTHP1)

SUBROUTINE FITON LOADS UP A MIDPT ARRAY OF TIMES AND POSITION, CALCULATES THE VELOCITIES IN TERMS OF RAW POSITION AND SMOOTHING COEFFICIENTS AND STORES THESE IN ARRAYS. ITS ARGUMENTS ARE:

ICNTPZ = INDEX INTO VELOCITY AND MIDPT ARRAYS WHERE DATA IS TO BE STORED.
LTHP1 = SMOOTHING INTERVAL LENGTH FOR CALCULATING VELOCITY.

INCLUDE 'RCOMMON.SA:1'

PX=0.0
FY=0.0
PZ=0.0
DO 1 I=1,LTHP1
  FZX=FZX+Z(I)*PXY1(I)
  PX=PX+X(I)*PXY1(I)
  FY=FY+Y(I)*PXY1(I)
1 TMID(ICNTP2)=TIM(NX1MID)
  XMID(ICNTP2)=X(NX1MID)
  YMID(ICNTP2)=Y(NX1MID)
  ZMID(ICNTP2)=Z(NX1MID)
7 XVM(ICNTP2)=PX*2
  YVM(ICNTP2)=PY*2
  ZXM(ICNTP2)=PZ*2
RETURN
END
SUBROUTINE FRQTMP (T, FREQ, TEMP, FAIL)

PURPOSE
Calculates the sensor temperature (TEMP) given a signal frequency (FREQ). The ratio between FREQ and the reference signal (calculated by a quadratic equation for time) is the search key for the table FCSOND. The sensor resistance RWNT is found by interpolation from table RCSOND. A second search of table RCBEAD gives the temperature. The search technique is binary.

DUMMY VARIABLES
FAIL Search-fail flag. FAIL=1 indicates that either FRATIO is outside the FCSOND table or RWNT is outside of the RCBEAD table so sensor temperature cannot be found. Otherwise FAIL=-1.
FREQ Sonde sensor signal frequency (Hertz) to be used to calculate sensor temperature
T Time after launch (seconds).
TEMP Sensor temperature (degrees Kelvin)

INTERNAL VARIABLES
IH, IL, IM Upper, lower, and mid indices used in table searches
RATIO Temporary storage variable
RWNT Sensor resistance calculated by linear interpolation.

COMMON VARIABLES
C(3) Coefficients for quadratic least squares fit to the reference signal
FCSOND(25) Sonde calibration data, frequency ratio associated with corresponding sensor resistance values in RCSOND.
FRATIO Ratio of sonde frequency to reference frequency
NCBEAD Number of calibration points for the thermistor given for the present flight.
NCSOND Number of calibration points for the sonde for the present flight.
RCBEAD(10) Thermistor calibration data, resistance values corresponding to thermistor temperatures in TCBEAD.
RCSOND(25) Sonde calibration data, sensor resistance values corresponding to frequency ratios in FCSOND.
TCBEAD(10) Thermistor calibration data, temperature (degrees C) corresponding to resistance values in RCBEAD.

CALLED BY - DRIVER

SUBROUTINES CALLED - NONE

TERMINATION CONDITIONS
If no values are found in FCSOND bracketing FRATIO, or no values are found in RCBEAD bracketing RWNT, TEMP is left undefined, the 'search-fail' flag is set to 1, and the routine returns. Otherwise, TEMP is calculated.
This routine is identical to the METROC-K version except:
1. The flag FAIL was added to indicate search failure as the RETURN statement is not available on this machine.
2. The storage variable RATIO was added because AMOD will not take arithmetic expressions as arguments.
3. The dummy variable TIME was changed to T to avoid conflict with a common variable.

COMPILED 8/23/83

THE DATA IN THE 'CALBRA' LABEL COMMON ARE READ IN BY THE PROGRAM 'MAIN' AND THEN ADJUSTED BY THE SUBROUTINE 'CLOBSET',

FAIL = -1.
FRATIO = TC(T*C(1) + C(2)) + C(3)
FRATIO = FREQ / FRATIO

SEARCH 'FCSOND' FOR VALUES BRACKETING 'FRATIO'

IM = NCSOND + 1
IL = 0
100 IH = IL + (IM-IL)/2
   IF (IM.EQ.IL) GO TO 500
   IF (FCSOND(IM) - FRATIO) 100,160,120
120 IF (IM.EQ.NCSOND) GO TO 500
   IF (FCSOND(IM+1) - FRATIO) 140,150,130
130 IL = IM
   GO TO 110

'RWNT' IS THE SONDE RESISTANCE CALCULATED BY LINEAR INTERPOLATION.

140 RWNT = RCSOND(IM) + (FRATIO = RCSOND(IM)) x
          (RCSOND(IM+1) = RCSOND(IM)) / (FCSOND(IM+1) - FCSOND(IM))
   GO TO 190
150 IM = IM +1
160 RWNT = RCSOND(IM)
190 RWNT = EXP(RWNT)

SEARCH 'RCBEAD' FOR VALUES BRACKETING RWNT

IM = NCBEAD + 1
IL = 0
200 IH = IM
210 IM = IL + (IM - IL) / 2
   IF (IM.EQ.IL) GO TO 500
   IF (RCBEAD(IM) - RWNT) 200,260,220
220 IF (IM.EQ.NCBEAD) GO TO 500
   IF (RCBEAD(IM+1) - RWNT) 240,250,230
230 IL= IM
   GO TO 210

'TEMP' IS CALCULATED BY AN INTERPOLATION SCHEME
C

240 RATIO = RCBEAD(IM) / RCBEAD(IM + 1)
    TEMP = ALOG (RATIO) / (1.0/TCBEAD(IM+1) - 1.0/TCBEAD(IM))
    RATIO = RCBEAD(IM) / RWNT
    TEMP = ALOG (RATIO) / TEMP + 1.0/TCBEAD(IM)
    TEMP = 1.0/TEMP
    RETURN
250 IM = IM + 1
260 TEMP = TCBEAD(IM)
    RETURN

C

SEARCH HAS FAILED

C

500 FAIL = 1.
    RETURN
END
FUNCTION GPTOM (Z)

C=============================================================================
C Converts geo-potential altitude to meters altitude
C=============================================================================

CONSTANTS
XX Equatorial radius squared in meters
ZZ Polar radius squared in meters
9.780356 Equatorial gravity constant in Lambert's Gravity Form.
9.8 One geo-potential unit = 9.8 geometric units
5.7704E-5 Geodetic constant in Lambert's Gravity Formula
5.1723913E-2 Geodetic constant in Lambert's Gravity Formula

DUMMY VARIABLES
Z Geo-potential altitude
GPTOM Altitude in meters

INTERNAL VARIABLES
IPASS Flag indicating the first time the function is called so that radius and gravity can be calculated. Values:
1=first call of the function
0=subsequent call

GPHE Lambert's Gravity Formula with altitude terms omitted
RADIUS Computed nominal earth radius in meters
SN Sine of the geodetic latitude of the radar site
SN2 Sine of twice the geodetic latitude of the radar site
LAT2 Twice the geodetic latitude of launch site in degrees

COMMON VARIABLES
XLAT Geodetic latitude of the launch site in degrees

CALLING ROUTINES - DRIVER

SUBROUTINES CALLED - ZSIN, ZCOS

REFERENCES

CHANGES
Identical to METROC-K version except:
1.Deleted unused constant GRAV
2.Added temporary storage variable LAT2.
3.Used new functions ZCOS and ZSIN because COS and SIN cannot take negative arguments

COMPILED 8/22/83

=============================================================================

DATA IPASS/1/
DATA XX/ 4.06801E13/, ZZ/ 4.04068E13/

ON FIRST CALL COMPUTE RADIUS AND GRAVITY AT STATION LATITUDE

IF (IPASS.EQ.0) GO TO 10
IPASS = 0
RADIUS = XX*ZZ/(XX*ZSIN(XLAT)**2 + ZZ*ZCOS(XLAT)**2)
RADIUS = SQRT (RADIUS)
LAT2 = 2. * XLAT
SN2 = ZSIN(LAT2)
SN = ZSIN(XLAT)
GPHE = 9.780356 + 5.1723413E-02 * SN * SN
     -5.7704E-05 * SN2 * SN2

CONVERT GEOPOTENTIAL ALTITUDE TO METERS

10 GFTOM = (RADUS*9.8*Z) / (GPHE*RADUS - 9.8*Z)
RETURN
END
FUNCTION GRVITY (Z)

PURPOSE
Computes gravity as a function of altitude Z and latitude of the observer XLAT.

DUMMY VARIABLES
Z Altitude in kilometers

LOCAL VARIABLES
GA Computed quadratic coefficient in gravity formula
GB Computed linear coefficient in gravity formula
GRVITY Acceleration of gravity as a function of altitude and latitude (meters/second squared)
GRAV Nominal acceleration of gravity.
IPASS Flag indicating the first time the function is called. Possible values:
1= first call, calculate constants
0= subsequent call, constants already calculated
RADSO Square of RADUS, temporary variable
RADUS Nominal earth radius (meters)
XX Equatorial radius squared in meters
ZZ Polar radius squared in meters

COMMON VARIABLES
XLAT Geodetic latitude of launch site in degrees

CALLED BY - WIND, DRIVER

CHANGES
identical to the METROC-K subroutine except:
1. Variable RADSO has been added to store intermediate results (6/24/83)

COMPILED 6/24/83

INCLUDE 'RCOMMON.SA:1'
DATA IPASS /1/
IF (IPASS .EQ. 0) GO TO 10
IPASS = 0
RADSO = XX*ZZ/(XX*SIN(XLAT)**2 + ZZ*COS(XLAT)**2)
RADUS = SQRT (RADSO)
GA = 3.0*GRAV/RADUS**2
GB = -2.0*GRAV/RADUS
10 GRVITY = Z*(Z*GA + GB) + GRAV
RETURN
END


SUBROUTINE KMNTAB (Z, CK1, CK2, CK3, CK4, TIME, OUT)

C Lookup into the Krumins' tables according to the
C altitude Z for bracketing values in the array CCZ.
C The four entries are interpolated linearly.

DUMMY VARIABLES
CK1,CK2,
CK3,CK4 Interpolated Krumins coefficients
OUT Out of range flag. Values:
1. = altitude is outside range of Krumins table
   -1. = in range, coefficients returned
TIME Time since launch (seconds)
Z Altitude (Km) of sonde

INTERNAL VARIABLES
IN,IL,IM Upper, lower, and mid index in binary table search
IX Second index of CC4. 1 = day, 2 = night
RATE Ratio of altitudes for linear interpolation

COMMON VARIABLES
CC1(60) Krumins coefficient K1, aerodynamic heating factor
   (coefficient of the square of airspeed)
CC2(60) Krumins coefficient K2 (seconds), dynamic lag factor
   (coefficient of rate of change of sensor temperature)
CC3(60) Krumins coefficient K3, radiation emission factor
   (coefficient of fourth power of sensor temperature)
CC4(60,2) Krumins coefficient K4 (degrees K), radiation
   (solar, longwave) and electric heating input term
   CC4(j,1) = for day
   CC4(j,2) = for night
CCZ(60) Tabulated altitudes (meters) of the Krumins
   coefficients CC1,CC2,CC3,CC4
DAYNGT Day-night flag. Values: 1. = sonde is in daylight
   2. = night
NKMN Number of altitude levels in the Krumins table

CALLED BY - KMNTMP, DRIVER

SUBROUTINE CALLED - SUNUP

TERMINATION CONDITIONS
If altitude is outside the Krumins table range, the
lookup fails and the routine returns with OUT=1.

CHANGES
Identical to METROC-K version except:
1. The flag OUT was added to indicate the out-of-range
   condition.

COMPILED 8/20/83

! INCLUDE 'RCOMMON.SA:1'
OUT = -1. ; clear out-of-range flag

THE SEARCH TECHNIQUE IS BINARY

IM = NKMN + 1
IL = 0
10 IH = IM
20 IM = IL + (IH-IL) / 2 
   IF (IM.EQ.IL) GO TO 100  ; if Z below bottom of table
   IF (Z - CCZ(IM)) 10, 60, 30
30 IF (IM.GE.NKMN) GO TO 100  ; if Z above top of table
   IF (Z - CCZ(IM+1)) 70, 50, 40
40 IL = IM
   GO TO 20
50 IM = IM + 1
60 RATE = 0.0
   GO TO 80

INTERPOLATION

70 RATE = -(CCZ(IM) - Z) / (CCZ(IM) - CCZ(IM+1))
80 CK1 = CC1(IM ) + RATE*(CC1(IM ) - CC1(IM+1 ))
   CK2 = CC2(IM ) + RATE*(CC2(IM ) - CC2(IM+1 ))
   CK3 = CC3(IM ) + RATE*(CC3(IM ) - CC3(IM+1 ))
   IX = 1
   CALL SUNUP (Z, TIME)
   IF (.NOT. DAYNGT) IX = 2
   CK4 = CC4(IM,IX) + RATE*(CC4(IM,IX) - CC4(IM+1,IX))
RETURN

OUT OF RANGE

100 OUT = 1.
   RETURN
END
SUBROUTINE KMNTMP (TAIR, CAERO, CDYN, CEMISS, CABSRB, AIRSPD)
CALL Y.MNTAS (ZS, CK1, CK2, CK3, CK4, TIME, OUT)
IF (OUT) GO TO 10
CAERO = -(CK1 * AIRSPD * AIRSPD)
CDYN = CK2 * TEMPD
CEMISS = CK3 * TEMP * TEMP
CABSRB = CK4
TAIR = TEMP + CAERO + CDYN + CEMISS + CABSRB
RETURN
C = OUTSIDE KRAMINS TABLE RANGE, NO TEMP COR APPLIED ....
10 CAERO = 999.9
CDYN = 999.9
CEMISS = 999.9
CABSRB = 999.9
TAIR = TEMP
RETURN
SUBROUTINE LEGNDR(NPTS, MPWR, CF, CFSQ, CF1, CF2, NDER)
DIMENSION CF(10, 99), CFSQ(10), CF1(10, 99), CF2(10, 99), EM(11, 50),
DM(11), C(10), SUM(11), BINOM(11, 11), EI(10), CON1(10, 10), CON2(10, 10).
DATA BINOM/1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
0., 6., 30., 90., 210., 420., 756., 1260., 1980., 2970.,
0., 0., -20., -140., -560., -1680., -4200., -9240., -19440., -34320.,
0., 0., 0., 70., 350., 1155., 3465., 90990., 210210.,
0., 0., 0., 0., -252., -2772., -16632., -72072., -252252., -756756.,
0., 0., 0., 0., 0., -3432., -51480., -411840., -2333760.,
0., 0., 0., 0., 0., 0., 12870., 218790., 196910.,
0., 0., 0., 0., 0., 0., 0., -48620., -923780.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 184756./
DATA CON1/0., 1., 1., 1., 1., 1., 1.,
0., 2., -6., 22., -100., 548., -3528., 26136., -219168., 2053152.,
0., 0., -18., 105., -675., 4872., -39396., 354372., -3518100.,
0., 0., 0., -40., 240., -2940., 27076., -269136., 2894720.,
0., 0., 0., 0., -75., 875., -9800., 112245., -1346625.,
0., 0., 0., 0., 0., -126., 1932., -27216., 379638.,
0., 0., 0., 0., 0., 0., -9., 196., -3022., -66150.,
0., 0., 0., 0., 0., 0., 0., 0., -288., 6760.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., -405.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 10. /
DATA CON2/0., 2., -6., 22., -100., 548., -3528., 26136., -219168., 2053152.,
0., 0., 6., -36., 210., -1350., 9744., -78792., 708792., -7036200.,
0., 0., 0., 12., -120., 1020., -8820., 81228., -6894160.,
0., 0., 0., 0., -20., -30., -3500., 39200., 449890., -5386500.,
0., 0., 0., 0., 0., 30., -630., 9660., -136800., 1898190.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 56., -2016., 48720.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 72., -3240.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 90.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 10. /
100 FORMAT(14, 11F11.7)
N=NPTS-1
MPM1=MPWR-1

C SETUP INITIAL VALUES
SUM(1)=0.
DM(1)=1.
MID=1+N/2
DO 1 J=1, 50
1 EM(1, J)=1.

C COMPUTE BASIC CONSTANTS FOR COEFFICIENTS:
DM(K)=N*(N-1) *(N-2) ...(N+1-K)
EM(K, I)=I*(I-1) *(I-2) ...(I+1-K)
DO 3 K=2, 11
MK=1-K
FAC=2+N-K
DO 2 I=1, 50
MUL=I+MK
2 EM(K, I)=MUL*EM(K-1, I)
3 DM(K)=FAC*DM(K-1)
C  REPLACE EM(K,I) BY EM(K,I)/DM(K)
   DO 4 K=2,11
   SUM(K)=0.
   DO 4 I=K,50
   4   EM(K,I)=EM(K,I)/DM(K)
C
C  COMPUTE FIRST HALF OF COEFFICIENTS I=1, MID AND TOTAL SUMS SQUARES
C
   COMPUTE FIRST DERIVATIVE COEFFICIENTS (FIRST HALF)
   DO 24 M=1,MPWR
      C(M)=1.
      DO 23 K=1,M
         C(M)=C(M)+EM(K,I)*BINOM(M,K)
      TEMP=C(M)
      IF (ABS(TEMP) .LT. 0.1E-15) C(M) = 0.
      23 CONTINUE
      24 CFS(M,I)=C(M)

C  GENERATE SECOND HALF OF COEFFICIENTS
   INDX=N+2-I
   DO 7 M=1,MPWR,2
      CF(M,INDEX)=-CF(M,I)
   DO 8 M=2,MPWR,2
      CF(M,INDEX)=CF(M,I)

C  GENERATE CORRECT SUMS OF SQUARES IN SINGLE PRECISION
   DO 9 M=1,MPWR
      CFSQ(M)=0.0
      DO 9 I=1,NPTS
      9   CFSQ(M)=CFSQ(M)+CF(M,I)*CF(M,I)
   IF (NDER .EQ. 0) GO TO 10
   DO 37 I=1,MID
      EI(M)=I**M
      EII=I-1
      EI(1)=EII
      DO 20 M=2,MPWR
      20   EI(M)=EII*EI(M-1)
      EM(M,I)=NCP(M) OF I
      EM(1,I)=1.
      DO 21 M=2,MPWR
      21   EM(M,I)=EM(M,I)+CON1(M,K)*EI(K-1)
      DO 22 M=1,MPWR
      22   EM(M,I)=EM(M,I)/DM(M+1)
C
C  COMPUTE FIRST DERIVATIVE COEFFICIENTS (FIRST HALF)
   DO 24 M=1,MPWR
      C(M)=1.
      DO 23 K=1,M
      C(M)=C(M)+EM(K,I)*BINOM(M,K+1)
      TEMP=C(M)
      IF (ABS(TEMP) .LT. 0.1E-15) C(M) = 0.
      23 CONTINUE
      24 CFS(M,I)=C(M)
C GENERATE SECOND HALF OF COEFFICIENTS
  INDX=N+2-I
  DO 25 M=1,MPWR,2
  25 CF1(M,INDX)=CF1(M,I)
  DO 26 M=2,MPWR,2
  26 CF1(M,INDX)=CF1(M,I)
  IF(NDER .EQ. 1) GO TO 10
  C EM(M,I)=NCPP(M) OF I
  EM(1,I)=0.
  EM(2,I)=2.
  DO 31 M=2,MPWR
  31 EM(M,I)=EM(M,I)+CON2(M,1)*EI(K-1)
  C EM(M,I)=NCPP(M)/DM(M+1)
  DO 32 M=1,MPWR
  32 EM(M,I)=EM(M,I)/DM(M+1)
  C COMPUTE SECOND DERIVATIVE COEFFICIENTS (FIRST HALF)
  DO 34 M=1,MPWR
  34 C(M)=0.
  DO 33 K=2,M
  33 C(M)=C(M)+EM(K,I)*BINOM(M,K+1)
  TEMP=D(M)
  IF(ABS(TEMP).LT.1E-15) C(M)=1.
  CONTINUE
  34 CF2(M,I)=C(M)
C GENERATE SECOND HALF OF COEFFICIENTS
  INDX=N+2-I
  DO 35 M=1,MPWR,2
  35 CF2(M,INDX)=-CF2(M,I)
  DO 36 M=2,MPWR,2
  36 CF2(M,INDX)=CF2(M,I)
  37 CONTINUE
  10 RETURN
END
**MAIN**  
MAIN PROGRAM FOR 'K' VERSION OF METROC SYSTEM OF PROGRAMS  

**NAME** - METROC-K  
**NUMBER** - 112395  

**ABSTRACT** - METROC-IK IS DESIGNED FOR TOTAL AUTOMATED DATA REDUCTION OF METEOROLOGICAL ROCKETSonde PARAMETERS. THE ONLY MANUAL INTERFACE REQUIREMENT IS THE INPUT OF SONDE CALIBRATION DATA, SUPPORTING RAuSOUNDsonde DATA, AND NECESSARY START AND STOP INFORMATION FOR THE INPUT TAPE. THE PROGRAM USES A METPASS 1 (1,1,2397) OUTPUT TAPE. THE DATA ARE EDITED IN A CLEAN-UP ACTIVITY AND PASSED THRU ADDITIONAL SMOOTHING AND FILTERING WHERE WIND, TEMPERATURE, PRESSURE, AND OTHER PARAMETERS ARE CALCULATED. THE PROGRAM ALSO AVERAGES THE VELOCITY COMPONENTS USING A 21 POINT AVERAGING SCHEME. THE DATA ARE REFORMATTED AND A BASELINE TIE-ON LEVEL (BETWEEN RAuSOUNDSONDE AND ROCKET DATA) IS FOUND.

**PURPOSE**  
ALL INPUT CARDS FOR METROC-K ARE READ IN THIS ROUTINE AND DISPLAYED FOR EASY VERIFICATION. SUBROUTINE EDIT IS CALLED TO PASS ONCE THROUGH THE APPARATUS: DATA, CHECK FOR VALIDITY, AND EDIT IT. DRIVER IS CALLED TO PASS THROUGH THE DATA AGAIN, CALCULATING METEOROLOGICAL DATA. METFORM IS CALLED TO AVERAGE VELOCITY COMPONENTS AND DETERMINE THE TIE-ON LEVEL.

**COMMON BLOCKS** - /CTIM/, /IFMOP/, /TEMPA/, /CDATA/, /CGATE/, /CALBRA/

**INPUTS**  
BPRES - 30 WORD ARRAY OF RAuSOUNDsonde PRESSSURes  
BTMP - 30 WORD ARRAY OF RAuSOUNDsonde TEMPERATURE (CELSiUS)  
CALD - SODE SENSOR FREQUENCY CALIBRATION DATUM  
DPF - 30 WORD ARRAY OF DEWPOINT  
HEADER - 14 WORD LAUNCH OPERATION LABEL  
HRADAR - RADAR SITE HEIGHT ABOVE SEA LEVEL IN METERS  
ICAL - NOT USED IN METROC-K  
IENT - SWITCH, 1 = CALIBRATION REQUIRED  
IFILE - FILE NUMBER FOR USE WITH MULTIFILE INPUT TAPES  
IFM - OPTION, 0 = CHANGE SLANT RANGE FROM FEET TO METERS  
IRSKIP - NUMBER OF INPUT RECORDS TO SKIP BEFORE PROCESSING  
IWIND - LOGICAL SWITCH, 1. = RADAR DATA ONLY, -1. = TEMP.  
IZTIME - LAUNCH TIME IN HOURS, MINUTES AND SECONDS (INTEGER)  
ILALT - 30 WORD ARRAY OF RAuSOUNDSONDE ALTITUDE IN DECAMETERS  
LDIR - 30 WORD ARRAY OF RAuSOUNDSONDE WIND DIRECTION IN DEGREES  
LSPD - 30 WORD ARRAY OF RAuSOUNDSONDE WIND SPEED (M/S)  
NCBEAD - NUMBER OF THERMISTOR CALIBRATION POINTS  
NCSOND - NUMBER OF SONDE CALIBRATION POINTS  
NPBL - NUMBER OF RAuSOUNDSONDE DATA POINTS  
RCBEAD - CALIBRATION DATA FOR SENSOR
RCSOND - SONDE CALIBRATION DATA
REF - SONDE REFERENCED FREQUENCY PERIOD IN SECONDS
TCBEAD - THERMISTOR CALIBRATION DATA (TEMPERATURE DEG. CELSIUS)
TSTART - TIME TO START PROCESSING DATA (SECONDS AFTER LAUNCH)
TSTOP - TIME TO STOP PROCESSING DATA (SECONDS AFTER LAUNCH)
XLAT - GEODETIC LATITUDE OF LAUNCH SITE IN DEGREES
XLONG - GEODETIC LONGITUDE OF LAUNCH SITE IN DEGREES

* OUTPUT
HEADER, IZTIME, TSTART, TSTOP, IRSKIP, HRADAR, XLAT, XLONG,
IWISE, RCSOND, FCSOND, REF, CALD, TCBEAD, RCBEAD - CARD IPUT
DATA PRINTED ON OUTPUT LISTING
HGATE - HALF-WIDTH OF SIGNAL GATE USED FOR TRACKING MET DATA
SIGLEV - INITIAL VALUE OF SIGNAL GATE USED TO ACQUIRE, FILTER,
AND TRACK MET DATA = 155.0
TZERO - GMT LAUNCH TIME IN SECONDS

* INTERNAL VARIABLES
BI - OCTAL VALUE FOR ALL BLANKS 0377777777777
DTOR - CONVERSION FACTOR FOR DEGREES TO RADIANS
MXBEAD - MAXIMUM NUMBER OF THERMISTOR SENSOR CALIBRATION POINTS
MXSOND - MAXIMUM NUMBER OF SONDE CALIBRATION POINTS

* INPJT FILES
05 CARD BCD METROC-K INPUT CARDS 1 TO 37
06 LISTING BCD METROC-K INITIALIZATION AND
CALIBRATION DATA

* SCRATCH FILES - NONE
* CALLING ROUTINES - NONE
* SUBROUTINES CALLED - ERROR, CLBSET, EDIT, LDEDIT, DRIVER, METFORM
* COMMENTS
IF CALIBRATION DATA IS NOT WITHIN BOUNDS OR FCSOND AND RCBEAD
VALUES ARE NOT INCREASING, AN ERROR MESSAGE IS PRINTED AND JOB
IS TERMINATED

* MODIFICATION HISTORY
PROGRAMMER DATE MODIFICATION
BILL SPEIDEL 9/02/80 *ADDED FILE SKIP OPTION (FC 21)
2/11/81 *READ FEET TO METERS CONVERSION
OPTION FROM INPUT CARD 4
*PRINT CARD 4 ON FILE 06
CHARLOTTE TETER 8/24/81 *CALL METFORM

COMMON /CTIM/ ICAL,IENT, IDN
COMMON /IFMOP/ IFM
COMMON /TEMPA/ DRTMP(100),DRTIM(100),NOTEMP,NTPT
COMMON /CDATA/ HEADER(14),TSTART,TZERO,TSTOP,IWIND,
HRADAR, XLAT, XLONG,
& NPBL,LALT(30),LSPD(30),LDIR(30),DPF′(30),BTMP(30),BPRES(30),ALTG,
& IRSKIP,LEV

PAGE 002 METROC .SA:1
COMMON /CGATE/ SIGLEV,HGATE
COMMON /CALERA/ FCSOND(25), RCSOND(25), TCBEAD(10), RCBEAD(10),
& NCEBEAD, NCSOND
DIMENSION IZTIME(3)
DATA MXBEAD /10/, MXSOND /25/
  DATA BI/0377777777777777/ 
DATA DTOR / .01745329/ 
DATA IFC /21/ 
C+	 *READ INPUT CARDS AND INITIALIZE
SIGLEV = 155.
HGATE = 5.
READ (5,400) HEADER, IZTIME, TSTART, TSTOP, IWIND, IRSKIP,
& HRADAR, XLAT, XLONG, IFILE
IF (IFILE.EQ.0) IFILE = 1
NFILE = IFILE - 1
TZERO = 60.*60.*IZTIME(1)+IZTIME(2)+IZTIME(3)
WRITE (6,500) HEADER, IZTIME, TZERO, TSTART, TSTOP,
& IRSKIP, NFILE, HRADAR, XLAT, XLONG, IWIND, SIGLEV, HGATE
C+	 * PERFORM FILE SKIP
REWIND (21)
IF (NFILE.LE.0) GO TO 5
CALL FILFSP (IFC,NFILE)
5 CONTINUE
C+	 CONVERT DEGREES TO RADIANS
XLAT = XLAT * DTOR
XLONG = XLONG * DTOR
IF(IWIND) GO TO 16
C+	 DATASONDE CALIBRATIONS DATA...
READ (5,201) IENT,ICAL,IDN,IFM
IF (IENT.EQ.1) GO TO 11
WRITE (06,5200)
GO TO 12
11 WRITE (06,5000)
12 IF (ICAL.EQ.1) GO TO 13
WRITE (06,5300)
GO TO 14
13 WRITE (06,5100)
14 IF (IFM.EQ.1) GO TO 15
WRITE (06,5400)
GO TO 160
15 WRITE (06,5500)
160 CONTINUE
IF(IENT) ,16,
READ(5,420) NCSOND, NCEBEAD
C+	 PRINT ERROR MESSAGE IF CALIBRATION DATA IS OUT OF BOUNDS OR FCSOND
& AND RCBEAD VALUES NOT INCREASING
IF (NCEBEAD.GT.MXEBEAD) CALL ERROR(4)
IF (NCSOND.GT.MXSOND) CALL ERROR(5)
IF (NCEBEAD.LE.1) CALL ERROR(6)
IF (NCSOND.LE.1) CALL ERROR(7)
WRITE (6,520) HEADER
WRITE (6,550)
DO 20 I = 1, NCSOND
  READ(5,431) RCSOND(I), REF, CALD
20 CONTINUE
FCSOND(I) = REF / AMAX1(CALD, 1.)
IF(CALD.EQ.0.) REF = BI
IF(CALD.EQ.0.) CALD = BI
WRITE(6, 560) RCSOND(I), FCSOND(I), REF, CALD
IF (I.EQ.1) GO TO 20
IF (FCSOND(I).GE.FCSOND(I-1)) CALL ERROR(9)
20 CONTINUE
WRITE(6, 530)
READ(5, 431) (TCBEAD(I), I = 1, NCBEAD)
READ(5, 431) (RCBEAD(I), I = 1, NCBEAD)
DO 10 I = 1, NCBEAD
WRITE(6, 540) TCBEAD(I), RCBEAD(I)
IF (I.EQ.1) GO TO 10
IF (RCBEAD(I).GE.RCBEAD(I-1)) CALL ERROR(8)
10 CONTINUE
16 CONTINUE
C
C READ INPUT BASE LEVELS IF AVAILABLE
READ(5, 501) NPF'EL, (LALT(I), LSPD(I), LDIR(I), DPP(I), BTMP(I), BPR'ES(I)
& , I = 1, NPF'EL)
IF (IWIND) GO TO 30
C
C MODIFY CALIBRATION DATA
CLBSET 'MODIFIES THE TCBEAD, RCBEAD, RCSOND, AND FCSOND
C ARRAYS FOR LATER USE BY THE SUBROUTINE 'FRQTMP'.
CALL CLBSET
C
C EDIT, SMOOTH AND FILTER METEOROLOGICAL ROCKE'TSONDE PARAMETERS
30 CALL EDIT
CALL UDEDIT(IWIND)
CALL DRIVER
CALL METFORM
STOP
C
201 FORMAT(4I1)
400 FORMAT (14A6 / 3(I2, 1X), 1X, 2F10.1, 4X, L1, 13X, I7 /
& F10.2, 2F10.4, I3)
410 FORMAT (2(F10.2, F10.4))
420 FORMAT (2I5)
430 FORMAT (30X, F10.1, F10.3)
431 FORMAT (5F10.5)
440 FORMAT (30X, F10.1, F10.4)
C
500 FORMAT (1H1, 19A6 //
& 19X, 'TIME OF LAUNCH' , ' , I2, 2(1H*, I2), 1HZ,
& 3X, F7.0, ' SECS' //
& ' START PROCESSING (SECS PAST TOL)' , ' , F6.1, ' SECS' //
& ' STOP PROCESSING (SECS PAST TOL)' , ' , F6.1, ' SECS' //
& 7X, 'INPUT TAPE RECORDS TO SKIP=' , I7 /
& 7X, 'INPUT TAPE FILES TO SKIP=' , I7 //
& 19X, 'RADAR ALTITUDE' , ' , F7.1, ' METERS' //
& 19X, 'RADAR LATITUDE' , ' , F7.2, ' DEGREES' //
& 18X, 'RADAR LONGITUDE' , ' , F7.2, ' DEGREES COUNTED POSITIVE WE'
& 'ST OF GREENWICH' //
& 19X, 'WIND DATA ONLY+' , ' , L1, /19X,'SIGLEV,HGATE=', 2F6.1)
501 FORMAT (I2/(3(I5,2I3,F4.1,F5.1,F6.1)))
515 FORMAT (  
& 5X, 'SONDE ALTITUDE# ', F7.1, ' TO ', F7.1 /  
& 15X, 'SONDE AIR PRESSURE# ', F7.1, ' TO ',  
& F7.1, ' MILLIBARS')  
520 FORMAT (1H1, 14A6 // 'CALIBRATION DATA')  
530 FORMAT ( // 'TEMP-CENTIGRADE RESISTANCE-K OHMS' //)  
540 FORMAT ( 4X, F7.1, 12X, F8.3)  
550 FORMAT(// 'RESISTANCE-K OHMS FREQ RATIO REF. FREQ. SIG.' &  
& ' FREQ.'//)  
560 FORMAT(4X,F8.1,8X,F8.4,2F14.4)  
5000 FORMAT (1H0,T20,'21 POINT CALIB, DATA REQUIRED')  
5100 FORMAT (1H0,T20,'CALIB, DATA IS PRESENT')  
5200 FORMAT (1H0,T20,'WIND DATA OR DATA FILE 08 REQUIRED')  
5300 FORMAT (1H0,T20,'CALIB, DATA NOT PRESENT')  
5400 FORMAT (1H0,T20,'SLANT RANGE FROM INPUT TAPE IS IN FEET')  
5500 FORMAT (1H0,T20,'SLANT RANGE FROM INPUT TAPE IS IN METERS')  
END
SUBROUTINE POLYFT (N)

C PURPOSE
C Calculates the coefficients for a polynomial least-squares fit by solving the matrix equation AC = B
C The elements of A are supplied by array BT, which is generated by EDIT along with array B.
C C is the solution vector. The technique used is Gaussian elimination followed by back solution.

C
C
C A(i,j) = Sum(freq**(2N-i-j)) i>0, j>0
C = n i=0, j=0
C B(i) = Sum(freq*time**(N-i))
C
C BT(i) =

DUMMY VARIABLES
N Number of coefficients in the fitted polynomial, equals one plus the degree of the polynomial

LOCAL VARIABLES
DL
I
IL
IXA
J
JX

COMMON VARIABLES
A(3,3)
B(3)
BT(3)
C(3)

CHANGES
Identical to the METROC-K version except:
1. Array A in the original routine, which was equivalent to common array BT, was replaced with BT.
2. Array U was renamed A.

REFERENCES

INCLUDE 'RCOMMON.SA:1'

ELIMINATION

IL = 0
DO 30 I = 1,N
IXA = 2*(N-IL)
DO 10 J = I,N

;transfer Ith row into matrix A

10 CONTINUE
IXA = IXA-1 ; from storage array BT
10 A(I,J) = BT(IXA)
   IF (IL.EQ.0) GO TO 30
   DO 20 J = 1,IL
   DL = A(J,I)/A(J,J) ; divide Jth row by pivot A(J,J)
   C
   B(I) = B(I)-DL*B(J)
   DO 20 JX = I,N
   20 A(I,JX) = A(I,JX)-DL*A(J,JX) ; subtract DL times Ith row
   IL = IL+1
C-------------------------------
C PACK SOLUTION
C-----------------------------------------------------------
40 I = N
50 IF (I.EQ.IL) GO TO 60
   C(IL) = B(IL)-C(I)*A(IL,I)
   I = I-1
   GO TO 50
60 C(IL) = B(IL)/A(IL,IL)
   IL = IL-1
   IF (IL.NE.0) GO TO 40
RETURN
END
SUBROUTINE RDRCRD (RDATA)

C------------------------------------------------------------
C PURPOSE
C Converts the azimuth, elevation, and range information
C in array RDATA into rectangular X, Y, Z coordinates.
C The coordinates are then stored back in array RDATA,
C (with X,Y,Z stored respectively in the azimuth, elevation,
C and range)
DUMMY VARIABLES
Azimuth, elevation, and range input from EDIT
and X, Y, Z coordinates passed back to EDIT:
RDATA(1) azimuth (degrees), X coordinate (meters)
RDATA(2) elevation (degrees), Y coordinate (meters)
RDATA(3) range (meters), Z coordinate (meters)
CONSTANTS
CRVETH Reciprocal of twice the earth's radius (1/meters)
DTOR Degrees to radians conversion factor
INTERNAL VARIABLES
AZR Radar azimuth angle in radians
CA Cosine of radar azimuth
CE Cosine of radar elevation
ELR Radar elevation angle in radians
R Horizontal (ground) range of sonde from radar
SA Sine of radar azimuth angle
SE Sine of radar elevation angle
COMMON VARIABLES
HRADAR Height above sea level (meters) of radar site
CALLING ROUTINE - EDIT
SUBROUTINES CALLED - ZSIN, ZCOS
REFERENCES
Conversion formula: F.L. Staffanson, "METROC-K Algorithms"
University of Utah Engineering College Report UTECMR 79-161
November 1979, pp.10-11
CHANGES
This subroutine is identical to METROC-K version
except ZSIN and ZCOS functions replace standard 6809
trig functions because they don't take neg.arguments
COMPILED 5/17/83
C------------------------------------------------------------
INCLUDE 'RCOMMON.SA:1'
DIMENSION RDATA(3)
DATA DTOR / .01745329/ ;degrees to radians conv. factor
DATA CRVETH / 7.8491112E-08/ ;reciprocal of twice the earth's
                                 ;radius
CA = ZCOS (AZR)
SA = ZSIN (AZR)
CE = ZCOS (ELR)

AZR = RDATA (1) X DTOR ;convert azimuth from degrees to radians
ELR = RDATA (2) X DTOR ;convert range from degrees to radians
CA = ZCOS (AZR)
SA = ZSIN (AZR)
CE = ZCOS (ELR)
SE = ZSIN (ELR)
R = RDATA (3) * CE
RDATA (1) = R * SA
RDATA (2) = R * CA
RDATA (3) = RDATA (3) * SE + R*R*CRVETH + HRADAR
END
SUBROUTINE REMOVI (D, EMTY)

C-------------------------------------
C  For documentation see ADD1
C-------------------------------------

INCLUDE 'RCOMMON.SA:1'
DIMENSION D(5)
EMTY=-1.
& IF (ITAIL1.EQ.IHEAD1) THEN
 EMTRY=1.
 RETURN
 END IF
 DO 10 I=1,5
 D(I)=GFILE(I,ITAIL1)
 10 CONTINUE
 ITAIL1=ITAIL1+1
 IF (ITAIL1.GT.LNGTH) ITAIL1=1
 END
SUBROUTINE REMOV2 (D, EMTY)

C-----------------------------------
C For documentation see ADD1
C-----------------------------------

INCLUDE 'RCOMMON.SA:1'
DIMENSION D(5)
EMTY=-1.
& IF (ITAIL2.EQ.IHEAD2) THEN
  EMTY=1.
  RETURN
END IF
DO 10 I=1,5
   D(I)=QFILE(I,ITAIL2)
10 CONTINUE
ITAIL2=ITAIL2+1
IF (ITAIL2.GT.LNGTHQ) ITAIL2=1
END
SUBROUTINE REWRT (INTRVL,DINERT)

C-------------------------------------------------------------
C PURPOSE
Replaces the frequency datum of the last INTRVL
edited data points in queue1 with an "inert" value,
DINERT. Rewrites that would overrun the beginning
of the queue are stopped at the beginning (ITALI).

--->
(empty)

queue 2 D E queue 1
(values D IHEAD2 ITAIL1 E (data points
from D E from
DRIVER) D E EDIT)

^ ^ ^

ITALI2 IHEAD1

DUMMY VARIABLES
DINERT Constant to replace the frequency value
in QFILE(2,j) in the points to be rewritten.
INTRVL Number of points in the queue to be rewritten,
starting at the last point added.

INTERNAL VARIABLES
I Index of points to be rewritten.
ISTART Start pointer. Contains index in queue of
 first point to be rewritten.
ISTOP Stop pointer. Contains index in queue of
 last point to be rewritten, which is the
 last point in the queue.

COMMON VARIABLES
IHEAD1 Head pointer for queue1. Points to location
 of next point to be added to queue1.
ITALI1 Tail pointer for queue1. Points to location
 of next point to be removed from queue1.
QFILE(5,2000) Storage queue containing edited data points
 from EDIT and meteorological values from
DRIVER.
QFILE(1,j)= time since launch(seconds)
QFILE(2,j)= frequency datum (Hertz)
QFILE(3,j)= X coordinate (meters)
QFILE(4,j)= Y coordinate (meters)
QFILE(5,j)= Z coordinate (meters)

CALLING ROUTINES - EDIT

COMMENTS
1. Subroutine ADD1 initializes the queue pointers
 used by all the other queue access routines;
ADD2, REMOVI, REMOV2, REWRT. Therefore ADD1
 must be called at least once before any of the
 others are called.
2. REWRT will only work properly if IHEAD1 is
 greater than ITAIL1. This should always be
 true in EDIT and should be checked if the
 initial values of the pointers are changed.
CHANGES
REWRT is a new routine and was not part of METROC-K.

COMPILED 6/20/82

REFERENCES

------
INCLUDE 'RCOMMON.SA:1'
IF (IHEA1.LT.ITAIL1) THEN ;check proper pnter condition
  WRITE (101,1000)
END IF
IF (INTRVL.LT.0) RETURN ;handles case of INTRVL=0.
ISTOP=IHEA1-1 ;stop at last edited pt
ISTART=ISTOP-INTRVL ;start INTRVL pts back.
IF (ISTART.LT.ITAIL1) ISTART=ITAIL1
DO 10 I=ISTART,ISTOP
  QFILE(2,I)=DINERT
10 CONTINUE
1000 FORMAT ('*** ERROR IN REWRT'/&
  'HEAD POINTER LESS THAN OR EQUAL TO TAIL POINTER')
END
SUBROUTINE SINGLE(IDEG1,NPTFT1,CP1,CV1,CA1)
C
  SUBROUTINE SINGLE CALCULATES SMOOTHING COEFFICIENTS, ITS ARGUMENTS ARE:
C	IDEG1 = THE DEGREE OF TIME SMOOTHING POLYNOMIAL
C	NPTFT1 = THE SMOOTHING INTERVAL LENGTH
C	CP1 = DATA COEFFICIENTS
C	CV1 = VELOCITY COEFFICIENTS
C	CA1 = ACCELERATION COEFFICIENTS

DIMENSION P1(10,99),PPI(10,99),PPP1(10,99),PSQ1(10),CV1(100),CA1(100),CP1(100)
C
C	CALL LEGNDR TO OBTAIN LEGENDRE POLYNOMIAL COEFFICIENTS FOR FIT
CALL LEGNDR(NPTFT1,IDEG1,F1,PSQ1,PPI,PPP1)
MID=NPTFT1/2+1
PFTFT1=NPTFT1
C
COMPUTE DATA,VEL, AND ACC COEFFS FOR FIRST DEG OF FIT
DO 20 I=1,NPTFT1
CP1(I)=P1(1,I)*P1(1,MID)/PSQ1(1)+1.0/PFTFT1
CV1(I)=P1(1,I)*PPI(1,MID)/PSQ1(1)
CA1(I)=P1(1,I)*PPP1(1,MID)/PSQ1(1)
IF(IDEG1.EQ.1) GO TO 20
DO 5 J=2,IDEG1
C	COMPUTE DATA,VEL, AND ACC COEFFS FOR HIGHER DEGS OF FIT
CP1(I)=CP1(I)+P1(J,I)*P1(J,MID)/PSQ1(J)
CV1(I)=CV1(I)+F1(J,I)*PP1(J,MID)/PSQ1(J)
CA1(I)=CA1(I)+P1(J,I)*PPP1(J,MID)/PSQ1(J)
5 CONTINUE
20 CONTINUE
RETURN
END
SUBROUTINE SUNUP (ALT, TIME)

C Uses latitude, longitude, altitude, and day of year to compute
C a sunrise and sunset time for the sonde's present position.
C It then compares the time corresponding to the sensor's
C present position to the sunrise/sunset times to determine
C whether the sensor is in day or night, and sets flag DAYNGT
C accordingly. Subroutine KMNTAB uses DAYNGT to apply the
C proper day or night Krumins' temperature correction.

C CONSTANTS
DTOR	 Degrees to radians conversion factor
364.242	 Number of days in a year
23.4438333 Earth's inclination in degrees
279.9346 Factor for asymmetry due to ellipticity
in computing solar declination
3600 Hours to seconds conversion factor

DUMMY VARIABLES
ALT	 Altitude in meters
TIME	 Elapsed time from launch in seconds

INTERNAL VARIABLES
A	 The sunrise/sunset solar elevation angle (radians)
COSD	 Cosine of the solar declination angle
D	 Day of the year in radians
SIND	 Sine of the solar declination angle
H	 Solar hour angle (in decimal hours). (this is also
the number of hours in half a "day")
IYR	 Year of launch decoded from header
MONTH	 Month of launch decoded from header
MTHDAY	 Launch day of month decoded from header
SNOON	 The time (in decimal hours) of meridian passage
or true solar noon.
TIMNOW	 Time corresponding to altitude of sensor (GMT
in seconds).
TSR	 Time of sunrise (GMT in seconds).
TSS	 Time of sunset (GMT in seconds).
TWODEE	 Two times the day of the year in seconds
XPHI	 Angle (in radians) which takes into account the
ellipticity of the earth.

COMMON VARIABLES
DAYNGT	 Flag that indicates whether sensor is in day
or night. Values: 1.=day, -1.=night
HEADER	 14 character launch operation label
TZERO	 Launch time (GMT in seconds)
XLAT	 Geodetic latitude of launch site in degrees
XLONG	 Geodetic longitude of launch site in degrees

CALLED BY - KMNTAB

FUNCTIONS CALLED - ZSIN, ZCOS

COMMENTS
1. Date of launch must start at character 15 of HEADER
   and be in form YYMMDD.
2. Longitude XLONG is in degrees counted positive
west of Greenwich.

3. All formulas for this routine were derived from NASA's Technical Memorandum TM-X-1646

CHANGES

Identical to METROC-K version except:
1. Changed variable DAYMNTH to MTHDAY to make it integer, and variable M to SNOON to make it real.
   Variable DAYEAR is real instead of integer.
2. Used ENCODE statement to help decode HEADER
3. Changed the two values of flag DAYNGT to 1. = day, -1. = night
4. Added temporary variable TWODEE
5. Used functions ZSIN and ZCOS because SIN and COS cannot take negative arguments.

==================================================================
INCLUDE 'RCOMMON.SAIL'
DIMENSION DAYEAR(12)

INITIALIZE

DATA DAYEAR /0.,31.,59.,90.,120.,151.,
& 181.,212.,243.,273.,304.,334./
DATA DTOR/0.01745329/

CALCULATE DAY OF YEAR

ENCOD 2000, HEADER
DECODE 1000, NYR, MONTH, MTHDAY
DAYNGT = 1.
D = DAYEAR(MONTH) + MTHDAY - 1
IF((MONTH .GT.2) .AND. (MOD(NYR,4) .EQ. 0) ) D = D + 1.0
D = (D * 360. / 365.242) * DTOR

COMPUTE SOLAR HOUR ANGLE

TWODEE = 2.*D
XPHI = (279.9348 + D/DTOR + 1.914827*ZSIN(D) - 0.079525*ZCOS(D)
& + 0.019938 * ZSIN(TWODEE) - 0.001620 * ZCOS(TWODEE)) * DTOR
SIND = 23.44383333*DTOR
SIND = ZSIN(SIND) * ZSIN(XPHI)
SIND = 1. - ABS(SIND)**2.
COSD = SQRT(SIND)
AA = (-1.76459 * ((ALT/1000.)**40795)) * DTOR
BB = (ZSIN(AA) - ZSIN(XLAT) * SIND)/(ZCOS(XLAT)*COSD)
IB = BB
IF (IB) 9,5,10
5 H = (ACOS(EE))/(15.*DTOR)

COMPUTE TRUE SOLAR NOON

SNOON = 12. + 0.123570 * ZSIN(D) - 0.004289 * ZCOS(D)
& + 0.153809 * ZSIN(TWODEE) + 0.060783 * ZCOS(TWODEE)
TIMNOW = TZERO + TIME

DETERMINE IF DATA OBTAINED IN LIGHT OR DARKNESS

TSR = (XLONG/(15.*DTOR) + SNOON - H) * 3600.
TSS = (XLONG/(15.*DTOR) + SNOON + H) * 3600.
IF (TIMNOW .LT. TSR .AND. TIMNOW .GT. TSS-86400.) DAYNGT = -1.
IF (TIMNOW .GT. TSS .AND. TIMNOW .LT. TSR+86400.) DAYNGT = -1.
9 CONTINUE
RETURN
10 CONTINUE
DAYNGT = -1.
RETURN
1000 FORMAT (14X, 3I2, 64X)
2000 FORMAT (14A6)
END
SUBROUTINE SWAP (SCOPY)

PURPOSE
Exchanges the 13 smoothed temperature and position variables generated by DIRSIT with the values stored in array SCOPY.

DUMMY VARIABLES
SCOPY(13) contains previous values of the 13 temperature and position variables below in the order listed.
SCOPY(1)=TIME ... SCOPY(13)=TEMPDD

COMMON VARIABLES
TIME time after launch of present data point (s)
XS smoothed east coordinate (m)
YS smoothed north coordinate (m)
ZS smoothed altitude (m)
TEMP smoothed sensor temperature (K)
XD east velocity component (m/s)
YD north velocity component (m/s)
ZD upward velocity component (m/s)
TEMPD first derivative of sensor temperature (K/s)
XDD east acceleration component (m/s^2)
YDD north acceleration component (m/s^2)
ZDD upward acceleration component (m/s^2)
TEMPDD second derivative of sensor temperature (K/s^2)

INTERNAL VARIABLES
SAV temporary storage

SUBROUTINES CALLED - NONE

CALLING SUBROUTINES - DRIVR1

LAST CHANGE 12/6/82

INCLUDE 'RCOMMON.SA:1'

SAV = TIME
TIME = SCOPY(1)
SCOPY(1) = SAV
SAV = XS
XS = SCOPY(2)
SCOPY(2) = SAV
SAV = YS
YS = SCOPY(3)
SCOPY(3) = SAV
SAV = ZS
ZS = SCOPY(4)
SCOPY(4) = SAV
SAV = TEMP
TEMP = SCOPY(5)
SCOPY(5) = SAV
SAV = XD
XD = SCOPY(6)
SCOPY(6) = SAV
SAV = YD
YD = SCOPY(7)
SCOPY(7) = SAV
SAV = ZD
ZD = SCOPY(8)
SCOPY(8) = SAV
SAV = TEMPD
TEMPD = SCOPY(9)
SCOPY(9) = SAV
SAV = XDD
XDD = SCOPY(10)
SCOPY(10) = SAV
SAV = YDD
YDD = SCOPY(11)
SCOPY(11) = SAV
SAV = ZDD
ZDD = SCOPY(12)
SCOPY(12) = SAV
SAV = TEMPD
TEMPD = SCOPY(13)
SCOPY(13) = SAV
END
SUBROUTINE TIMCHK (DSTOR,NW,NERR)

C--
C PURPOSE
C Checks for errors in successive time values of raw data.
C Corrects the time of the point in error if there are less
C than a maximum number of errors in the given set of points.
C When the maximum is exceeded, the program aborts.

C DUMMY VARIABLES
C DSTOR(5,10) tenth second raw data record.
C DSTOR(i,j) time in GMT seconds
C NW number of raw data points in DSTOR to be checked
C for time errors. (NW=5 for first call to TIMCHK,
C NW=10 thereafter)
C NERR maximum number of time errors allowed before
C program stops

C INTERNAL VARIABLES
C DELT difference in time value between two successive
C raw data points
C IDSTOR temporary storage for time, used to truncate
C to the tenths place.
C IERFG error counter. If DELT < .5 or DELT > 1.5
C the counter is incremented.
C JL index of raw data points

C COMMON VARIABLES
C IO1 Output device number for informative messages
C Possible Values:
C 99 Dummy device. No output occurs
C 101 Terminal display screen
C 102 Printer

C CALLED BY - EDIT

C TERMINATION CONDITIONS
C If number of time errors during current call is greater
C than NERR: error message is printed and program stops

C COMMENTS
C A single raw data point which is merely missing,
C rather than wrong, will cause a chain reaction which
C will abort the program.

C CHANGES
C This subroutine differs from the METROC-K version:
C 1. All references to radar time were eliminated,
C since radar time is not present
C 2. The parameter NERR was added because eliminating radar
C may require changes in the error limit
C 3. Printing the error message and terminating are done
C directly by TIMCHK rather than by ERROR.

C COMPILED 6/25/83

INCLUDE 'RCOMMON.SA:1'
DIMENSION DSTOR(5,10)
IERFG = 0
DO 1 JL=2,NW
DELT = DSTOR(1,JL) - DSTOR(1,JL-1) ;difference should be .1 sec
IF (DELT.GT..15 .OR. 
  & DELT.LT..05) THEN ;if difference is out of
  IERFG = IERFG + 1 ; normal range,
  IF(IERFG.GT.NERR) THEN ; if too many errors in this set
    WRITE (IO1,1000) NERR, DSTOR(1,JK) ; print error message
    STOP ; and abort
  END IF
  DSTOR(JK,JL) = DSTOR(JK,JL-1) + .1 ; otherwise, correct error
  IDSTOR = (DSTOR(JK,JL) + .05) * 10.
  DSTOR(JK,JL) = IDSTOR / 10.
END IF
1 CONTINUE
1000 FORMAT ('*** ERROR IN TIMCHK'/
  & 'MORE THAN ',I2,' TIME ERRORS'/
  & 'LAST TIME CHECKED = ',F8.3,' SECONDS')
END
SUBROUTINE TMPFIT(IDRSIT,TIN,T,TOUT,TVOUT,TAOUT,ICNTT,LHT,T,ISLIDE)

C
C THIS SUBROUTINE USES A LINEAR FIT FOR THE TEMPERATURE, BUT IT
C USES A QUADRATIC FIT FOR THE FIRST AND SECOND DERIVATIVES.
C THE T1 AND T2 ARRAYS MUST BE DIMENSIONED AT LEAST LHT, WHILE
C THE TEMP AND TIME ARRAYS MUST BE DIMENSIONED BY AT LEAST (LHT +
C NSKIP). (NSKIP IS DEFINED IN SUBROUTINE DIRSIT).

C
C DIMENSION TEMP(100),P1(100),P2(100),TIME(100)
GO TO (10,20,20,60) IDRSIT

10 CONTINUE
DELT=0.5

C DEFINE TIME INCREMENT
ITEAR=(LHT+1)/2

C CALCULATE MIDPOINT OF SMOOTHING INTERVAL
P2SOR=LHT

C INITIALIZE
P1SOR=0.0
P2SOR=0.0

C CALCULATE THE COEFFICIENTS (ALONG WITH THE SUMS OF THEIR
C SQUARES) TO BE USED IN THE QUADRATIC FIT FOR THE DERIVATIVES.
DO 5 I=1,LHT
P1(I)=I-ITEAR
P2(I)=(I-ITEAR)**2 -(LHT**2-1.0)/12.0
P1SOR=P1SOR+P1(I)**2
P2SOR=P2SOR+P2(I)**2
5 CONTINUE
RETURN

20 CONTINUE

C PUT RAW TEMPERATURE AND TIME IN THE TEMP AND TIME ARRAYS(RESP)
TEMP(ICNTT)=T
TIME(ICNTT)=TIN
RETURN

60 CONTINUE

C PUT RAW TEMPERATURE AND TIME IN THE TEMP AND ARRAYS
TEMP(ICNTT)=T
TIME(ICNTT)=TIN

C CALCULATE SUMS NECESSARY FOR THE FIT OF TEMPERATURE AND ITS
C DERIVATIVES.
P0XSM=0.0
P1XSM=0.0
P2XSM=0.0
DO 80 I=1,LHT
P0XSM=P0XSM+TEMP(I)
P1XSM=P1XSM+(P1(I)*TEMP(I))
P2XSM = P2XSM + (P2(I) * TEMP(I))

80 CONTINUE

C CALCULATE SMOOTHED TEMPERATURE AND ITS DERIVATIVES.
TXOUT = P0XSM / P0SQR
TVOUT = P1XSM / (DELT * P1SQR)
TAOUT = 2.0 * P2XSM / (P2SQR * DELT * DELT)

C CORRESPONDING TO THE SMOOTHING INTERVAL MIDPOIN.
T = TEMP(ITE:AR)
TOUT = TIME(ITBAR)

C SLIDE THE DATA IN THE TEMP AND TIME ARRAYS.
IA = ICNTT - ISLIDE
DO 90 I = 1, IA
   IC = I + ISLIDE
   TEMP(I) = TEMP(IC)
   TIME(I) = TIME(IC)
90 CONTINUE
RETURN
END
FUNCTION ZCOS(A)

THE 6809 CANT TAKE COS OF A NEG #

DUMMY=ABS(A)
ZCOS=COS(DUMMY)
RETURN
FUNCTION ZSIN(A)

THE 6809 WILL NOT TAKE THE SIN OF A NEG #
THIS FUNCTION TAKES CARE OF THE PROBLEM

IF(A .GE. 0.0) GOTO 10
A = -A
ZSIN = SIN(A)
ZSIN = -ZSIN
RETURN
10 ZSIN = SIN(A)
RETURN
COMMON A(3,3)
COMMON ALTG
COMMON B(3)
COMMON BPRES(30)
COMMON ET(5)
COMMON ETMP(30)
COMMON C(3)
COMMON C1
COMMON CC1(60)
COMMON CC2(60)
COMMON CC3(60)
COMMON CC4(60,2)
COMMON CCZ(60)
COMMON DAYNGT
COMMON DFILE(5,600)
COMMON DM(185)
COMMON DMH
COMMON DMY(183)
COMMON DPP(30)
COMMON DRTEMP(100)
COMMON DRTIM(100)
COMMON DUM(184)
COMMON DUMMY(200)
COMMON FCS(25)
COMMON FRAI
COMMON FRKID
COMMON FRKID
COMMON HEADER(14)
COMMON HGATE
COMMON HRA
COMMON ICAL
COMMON IDEN
COMMON IDFSK
COMMON IDN
COMMON IENT
COMMON IFM
COMMON IMODEL
COMMON IOUT
COMMON IRSKIP
COMMON IWIN
COMMON LALT(30)
COMMOM LDIR(30)
COMMON LEV
COMMON LSPD(30)
COMMON NAVG
COMMON NCBEAD
COMMON NCSOND
COMMON NKM
COMMON NKPT
COMMON NOTEMP
COMMON NPBL
COMMON NTPT
COMMON NX1mid
COMMON PXY1(100)
COMMON PXY2(100)
COMMON RCBEAD(10)
COMMON RCSOND(25)
COMMON SCOPY(13)
COMMON SFR(100)
COMMON SIGLEV
COMMON ST1(100)
COMMON SX(100)
COMMON SY(100)
COMMON SZ(100)
COMMON TAPOG
COMMON TCBEAD(10)
COMMON TEMP
COMMON TEMPD
COMMON TEMPDO
COMMON TIME
COMMON TMID(50)
COMMON TMPTRS(1150)
COMMON TSTART
COMMON TSTOP
COMMON TZERO
COMMON U(3,3)
COMMON XD
COMMON XDD
COMMON XLAT
COMMON XLONG
COMMON XMID(50)
COMMON XS
COMMON XVM(50)
COMMON YD
COMMON YDD
COMMON YMID(50)
COMMON YS
COMMON YUM(50)
COMMON ZAPOG
COMMON ZD
COMMON ZDD
COMMON ZMID(50)
COMMON ZS
COMMON ZX(50)