(NASA-CR-175358) A PORTABLE REAL-TIME DATA PROCESSING SYSTEM FOR STANDARD METEOROLOGICAL RADIOSONDES Final Technical Report, 15 May 1981 - 30 Nov. 1982 (Utah Univ.) 240 p HC A11/MF A01 CSCL 04B G3/47

N84-16719

Unclas 15038

UMET-1

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. Staffanson
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

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.

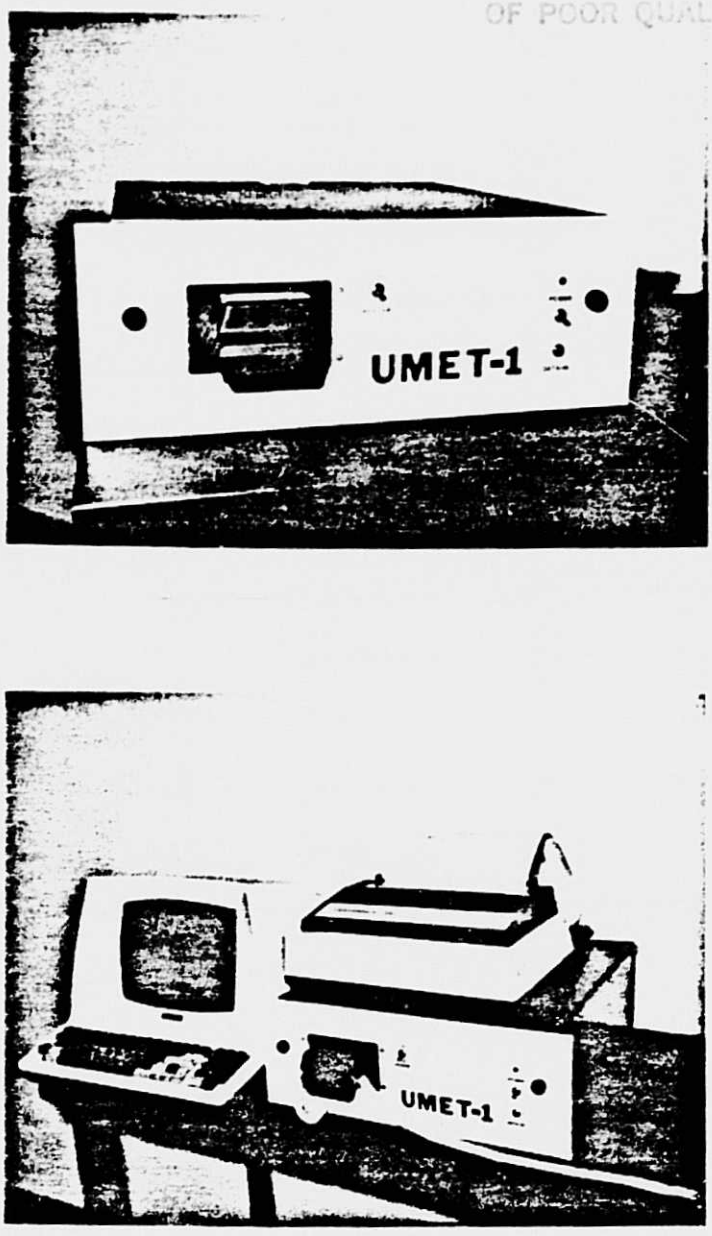


Fig. 1.0-1. UMET consists of the three portable units: the central unit (containing the printed-circuit boards, a paper tape reader, and power supply), a keyboard-video terminal, and a printer. Real-time data are received from the TRADAT system at the "DATA-IN" jack on the front panel. Baro-switch calibration 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 C1 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

Switch S1	Switch S2
1 D	1 D
2 D	2 D
3 D	3 U
4 D	4 D
5 D	5 D
6 D	6 U
7 D	7 U
8 D	8 D
9 U	9 U
10 D	10 D
(U = up, D = down)	

<table>
<tr><td colspan="2">Interrupt Vectors</td><td>$FFFF
$FFF2</td></tr>
<tr><td colspan="2">Monitor and
Interrupt Error Message</td><td>$FA37
$F800</td></tr>
<tr><td colspan="2">Terminal ACIA</td><td>$EC15
$EC14</td></tr>
<tr><td colspan="2">Printer Interface</td><td>$EC13
$EC10</td></tr>
<tr><td colspan="2">(unused)</td><td></td></tr>
<tr><td colspan="2">Paper Tape Reader PIA</td><td>$E017
$E014</td></tr>
<tr><td colspan="2">Real-Time Data PIAs</td><td>$E013
$E000</td></tr>
<tr><td colspan="2">RAM</td><td>$DFFF
$8000</td></tr>
<tr><td>ROM bank A
(AMET)</td><td>ROM bank B
(BMET)</td><td>$7FFF
$0000</td></tr>
</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 M68MMLC1 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, $\overline{\text{BIT6}}$) 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,

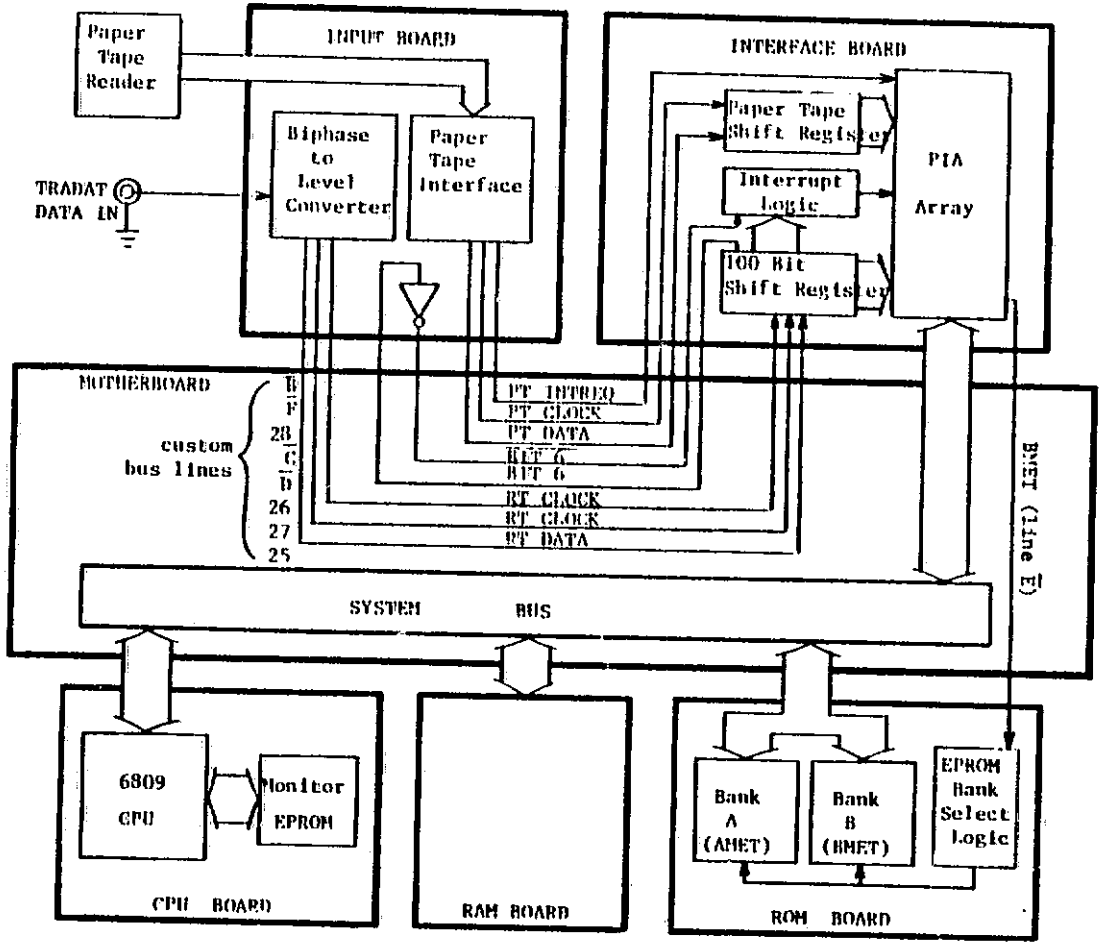


Fig. 2.0-1. The hardware configuration of UMET-1 processor. Eight data lines are used from the Input to the Interface Board, and one from the Interface Board to the ROM Board. Addresses $0000 to $7FFF, and $8000 to $DFFF are contained in the ROM and RAM Boards, respectively.

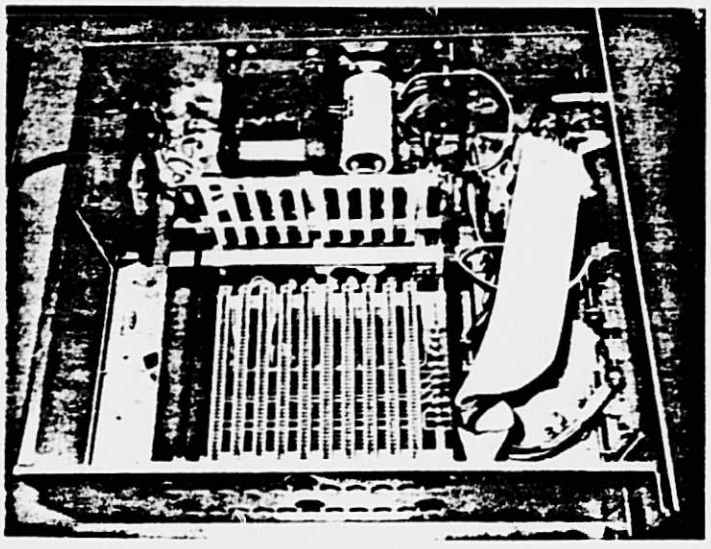


Fig. 2.1-1. The card-cage and motherboard. Position "1" of the card-cage is nearest the chassis front panel (right). Notice the custom connections hardwired on the motherboard between the headers at positions 7, 8, and 10. Cable C3, which connects the front panel to the Input Board (at edge connector 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

Position	Printed Circuit Card or Board
1-6	CPU, RAM BOARDS
7	ROM BOARD
8	INTERFACE BOARD
9	(unavailable position)
10	INPUT BOARD

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 $\bar{C}$, $\bar{D}$, $\bar{E}$, $\bar{F}$, 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 $\bar{B}$) 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 baroswitch 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

Input Board Pin	Interface Board Pin	ROM Board Pin	Signal Names (Fig. 2.0-1)	Function
25	25		RT DATA	Real-time data
26	26		RT CLOCK	Real-time clock
27	27		RT CLOCK	Real-time clock
$\bar{D}$	$\bar{D}$		BIT6	Real-time bit 6
$\bar{C}$	$\bar{C}$		$\overline{\text{BIT6}}$	Real-time bit 6, inverted
28	28		PT DATA	Paper tape data
$\bar{F}$	$\bar{F}$		PT CLOCK	Paper tape clock
$\bar{B}$	$\bar{B}$		PT INTREQ	Paper tape interrupt request
	$\bar{E}$	$\bar{E}$	BMET	ROM bank select

tape, and nine phototransistors to sense the light. The current through the LED array is adjusted to bias the phototransistors into full conduction 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.

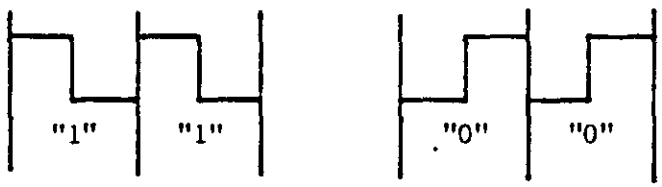
2.3 Input Board

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.

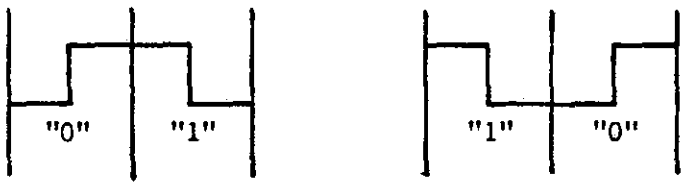
2.3.1 Biphase-to-level Converter

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 it's 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 generates 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 sequen-



ces 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 threshhold 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 74LS123 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 ($\overline{CH1}$ - $\overline{CH8}$) and one sprocket hole signal ($\overline{SPROCKET}$), 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.

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 $\bar{F}$, labelled PT CLOCK. At the conclusion of the eight pulses an interrupt is sent to the PIA through custom bus line $\bar{B}$, 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. $\overline{CS}$ is also on the header along with A0-A15 and $\overline{A0}$-$\overline{A15}$.

The address switches are set so that the base address of the board is $5000. Address lines 0 through 4 are cut so that the chip select (CS) pin on the board at wire-wrap header K1 pin 7, ($\overline{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, $\overline{CS2}$. 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, RS0 and RS1. All PIA's are wired so A0 goes to RS1 and A1 goes to RS0. 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

	A15 - A5	A4	A3	A2	A1 A0
PIA #1	Decoded by	0	0	0	
PIA #2	switch as	0	0	1	
PIA #3	CS ($E000)	0	1	0	register
PIA #4	on the wire	0	1	1	selects
PIA #5	wrap header,	1	0	0	
PIA #6	K1	1	0	1	

TABLE 2.4.1-2. PIA CHIP SELECT INPUTS

PIA #	CS0	CS1	$\overline{CS2}$
1	$\overline{A4}$	$\overline{A2A3}$	$\overline{CS}$
2	$\overline{A4}$	$A2\overline{A3}$	$\overline{CS}$
3	$\overline{A4}$	$\overline{A2}A3$	$\overline{CS}$
4	$\overline{A4}$	A2A3	$\overline{CS}$
5	A4	$\overline{A2A3}$	$\overline{CS}$
6	A4	$A2\overline{A3}$	$\overline{CS}$

TABLE 2.4.1-3. PIA REGISTER SELECTION

A1	A0	Register Selected
0	0	PA (Peripheral Register A)
0	1	PB (Peripheral Register B)
1	0	CRA (Control Register A)
1	1	CRB (Control Register B)

hex address	PIA	register	- register bits - 8 7 6 5 4 3 2 1
E000	PIA 1	PA	met word-d4 | met word-d3
E001		PB	met word-d2 | met word-d1
E002		CRA	(ROM bank switching)
E003		CRB	
E004	PIA 2	PA	azimuth-d3 | azimuth-d2
E005		PB	azimuth-d1 | met word-d5
E006		CRA	
E007		CRB	
E008	PIA 3	PA	elevation-d2 | elevation-d1
E009		PB	(empty) | az-d5 | azimuth-d4
E00A		CRA	
E00B		CRB	
E00C	PIA 4	PA	seconds-d2 | seconds-d1
E00D		PB	elevation-d4 | elevation-d3
E00E		CRA	
E00F		CRB	
E010	PIA 5	PA	hrs-d2 | hours-d1 | min-d2
E011		PB	min | minutes-d1 | secs-d3
E012		CRA	
E013		CRB	
E014	PIA 6	PA	
E015		PB	paper tape character
E016		CRA	
E017		CRB	

Fig. 2.4.1-1. PIA memory map. 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 2.4.2-1. TRADAT 100-BIT REAL-TIME DATA WORD FORMAT

	word length	bit number
Frame sync	8	1- 8
Spare	1	9
Met flag	1	10
Spare	1	11
Tracker mode	1+1+1+3+3	12- 20
Time of day (hr, 2 digit BCD)	2+4	21- 26
(min, 2 digit BCD)	3+4	27- 33
(sec, 2 digit BCD)	3+4	34- 40
(tenth sec, BCD)	4	41- 44
Elevation (deg, 4 digit BCD)	4+4+4+4	45- 60
Spare	1+1	61- 62
Azimuth (deg, 5 digit BCD)	2+4+4+4+4	63- 80
Met word (μsec, 5 digit BCD)	4+4+4+4+4	81-100
(or range)		

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

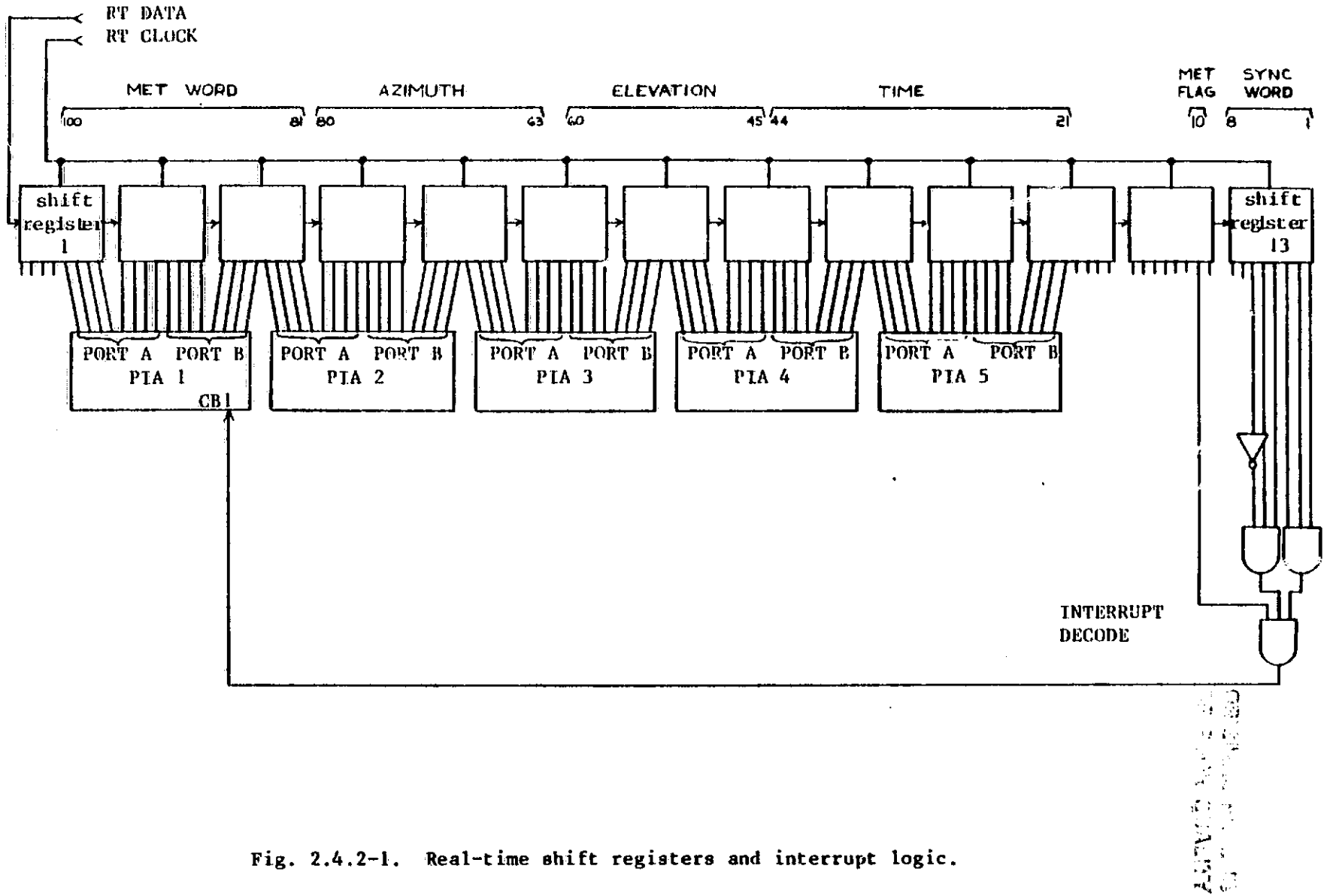


Fig. 2.4.2-1. Real-time shift registers and interrupt logic.

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 CA1 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 $\overline{A15}$ 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

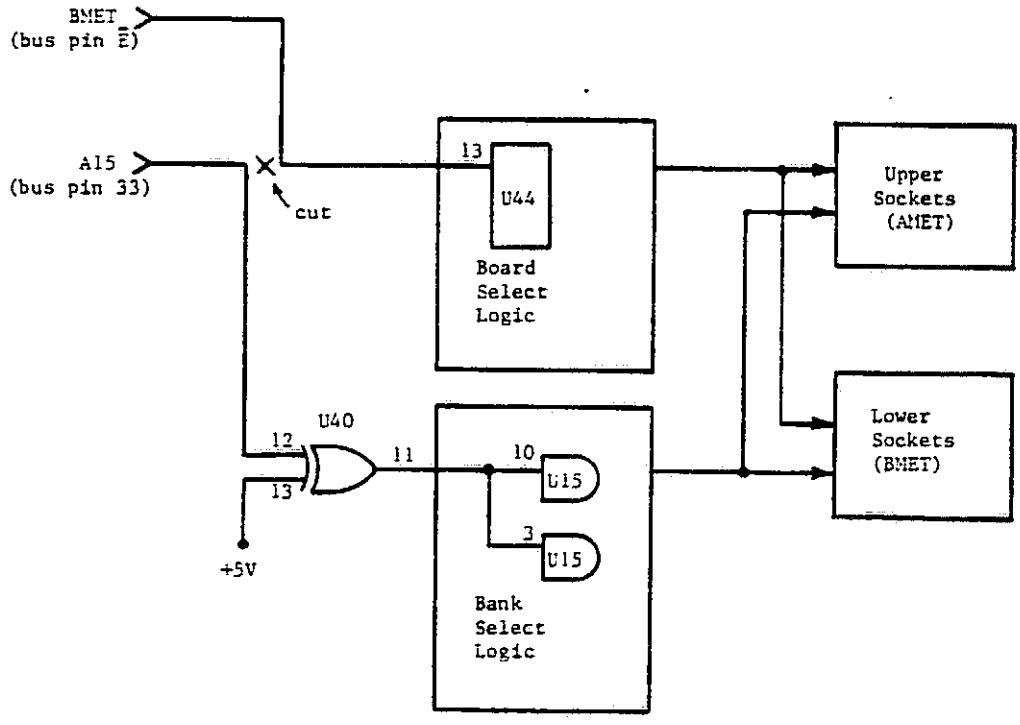


Fig. 2.5-1. ROM Board modification for switching banks.

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 $\overline{A14}$. Row B is selected using A15 by way of board select, A14, and $\overline{A13}$.

2.7 Central Processor Unit (CPU) Board

The CPU Board is a Motorola M68MM19A Monoboard Microcomputer Micromodule 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.)

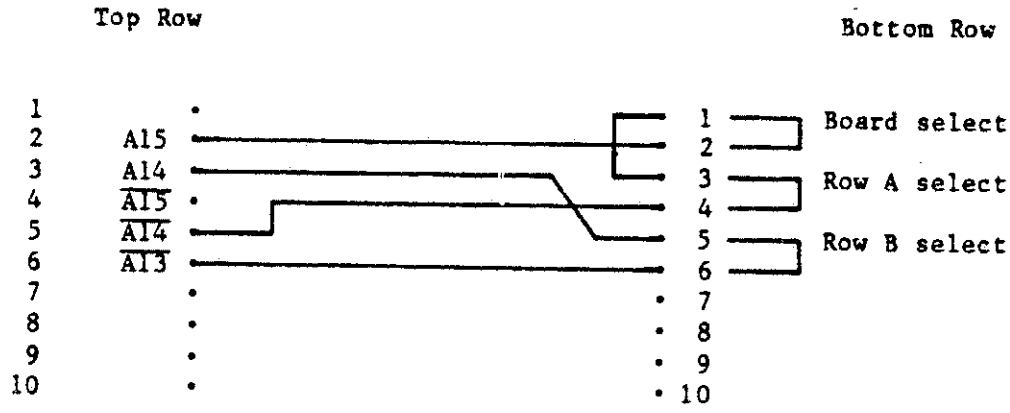


Fig. 2.6-1. RAM address select header, 24k.

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 $F831.

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.

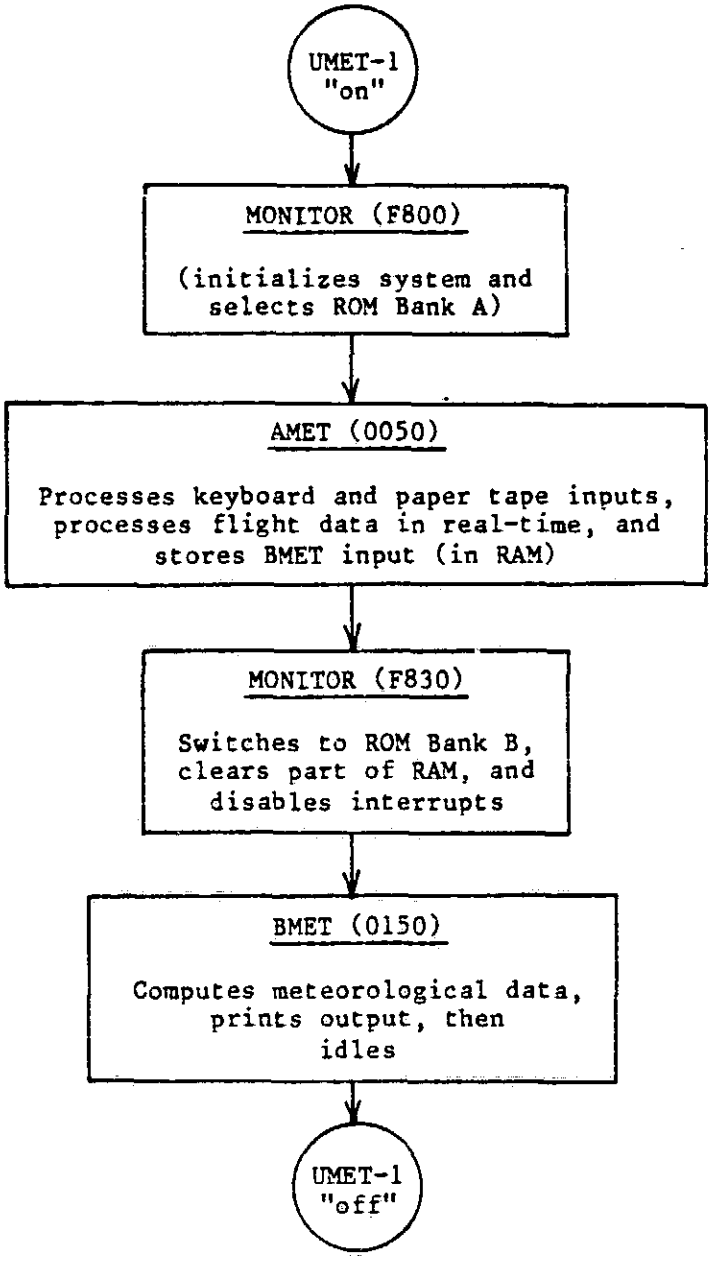


Fig. 3.0-1. Overall sequence of the UMET-1 program.

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

INTERRUPT LINE	INTERRUPT VECTOR	PSEUDO VECTOR	ROUTINE CALLED
RESET	$F800		Beginning of Monitor
IRQ	$DFFD, A =	$5878	TAPE1
		$5620	REAL
		$F900	Interrupt Error Message

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, $FFF8 (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 baroswitch) 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

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- ime 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 $DFFF. The uncleared RAM contains COMMON memory holding the input for BMET. Next, since the system will need no more input interrupts, MONITOR in effect disables subsequent IRQ interrupts. This is done by storing the starting address of the Interrupt Error Message routine ($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

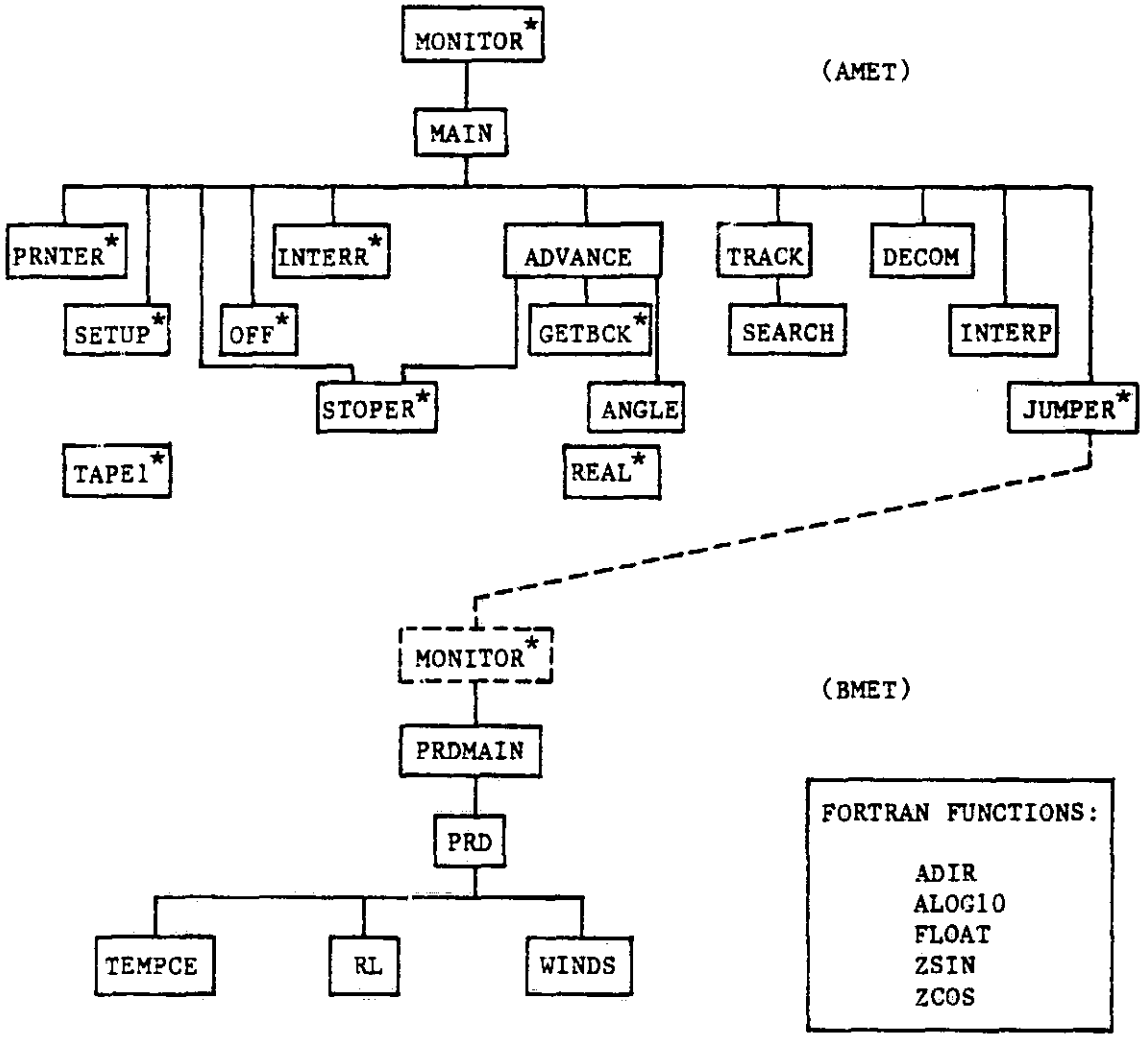


Fig. 3.3-1. Software routines of UMET-1. IRQ interrupts are masked by MONITOR until after SETUP, are then serviced by TAPE1 until INTERR, then by REAL until the end of AMET. Otherwise, all interrupts cause an Interrupt Error Message. AMET processes input in real-time, BMET computes output at the end 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 $5B52) [13, p.E-5]. This initializes IOPKG for use. The IOPKG subroutine LOUTC$ (location $5C0A) is called with the character to be sent, ESC ($1B), in the A register. This sends ESC to the printer. Then LOUTC$ 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 "3"). 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 ($\overline{IRQA}$) 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 $\overline{IRQ}$ 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 PIAs. The X register is used to address the PIAs. 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 $\overline{IRQ}$ is unmasked by clearing bit "I" in the condition

* 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 $\overline{IRQ}$ 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 awating 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, MASK=1). Processing is terminated also when successful, if the undecommutated array exceeds 900 time points (noisy data, MASK=2).

3.3.14 DECOM

Subroutine DECOM determines for each condensed point, COND, it's proper channel (temperature, reference, high reference, or relative humidity), and for each reference and humidity switch point, it's 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.

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 computed. 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 sonde performance.

After each reference point is processed, the terminal displays the progress of the automatic adjustment of the reference frequency thresholds RFL and PFL [2]. Sonde frequencies above RFL are identified as reference signal -- those above PFL are distinguished as high reference. Since sondes differ in reference frequency, and drift in frequency during flight, UMET automatically adjusts RFL and PFL by "tracking" 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 D), 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

1. ECC-PRD, NASA Computer Program 3.0.0700, NASA Wallops Computer Program Abstracts Vol.XXVII

2. "RAWINPROC - Computer Program for Decommutating, Interpreting, and Interpolating RAWINSONDE Meteorological Balloon Sounding Data", Final Report under NASA Research Grant NAG 6-8, University of Utah College of Engineering Report UTEC MR81-031, University of Utah, Salt Lake City, Utah 84112, February 1981

3. "Model 739-1/739-2 Printer Operator's Manual",37400911 Rev A, Centronics Data Computer Corporation, Hudson, NH 03051

4. "Model 910 Operator's Manual" No.B30000 3-001, Televideo Systems Inc., 1170 Morse Avenue, Sunnyvale, CA 94086, May 1981

5. "Micromodule Enclosure and System Hardware Chassis, Rack Mounting Kits, Card Cages, and Power Supply", M68MMESH(D2), Motorola Inc., October 1980

6. "RP-9362/RPF-9362 Tape Reader/Punch," EECO (Electronics Engineering Company of California), 1441 East Chestnut Avenue, Santa Ana, California 92701, Doc. No. 11142, October 1978.

7. "Reference Data for Radio Engineers," Fifth Edition, Howard W. Sams & Co. Inc., New York, (International Telephone and Telegraph Corporation), 1970, p. 32-23.

8. "MEX68USM Universal Support Module User's Guide", MEX68USM(D), Motorola Inc., August 1978

9. "TRADAT/MTTS System Instruction Manual", Oklahoma State University Electronics Laboratory, Stillwater, OK 74078, June 1980

10. "M68MM04A ROM/EPROM Module Micromodule 4A User's Guide", M68MM04A(D2), Motorola Inc., December 1980

11. "16K/64K Dynamic RAM Module User's Guide", MEX6864(D2), Motorola Inc., March 1979

12. "M68MM19/19A Monoboard Microcomputer Micromodule 19/19A", M68MM19(D2), Motorola Inc., March 1980

13. "MDOS FORTRAN Reference Manual", M68FTN(D3), Motorola Inc., September 1980

14. "MC6821 Peripheral Interface Adapter", data sheet DS9435R1, Motorola Inc., 1978

15. "MC6809,MC68A09,MC68B09 8-Bit Microprocessing Unit", data sheet DS9845, Motorola Inc., 1981

APPENDIX A

Program Listings

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.

```
00001                                   OPT     L
00002 F800                              ORG     $F800
00003 F800 1A   FF                      ORCC    #$FF    ;disable IRQ interrupt
00004 F802 86   81                      LDA     #$81    ;load direct page
00005 F804 1F   8B                      TFR     A,DP    ; register with $81 (?)
00006 F806 2                            NOP
00007 F807 12                           NOP
00008 F808 12                           NOP
00009 F809 12                           NOP
00010 F80A 86   03                      LDA     #$03    ;reset the ACIA
00011 F80C B7   EC14                    STA     $EC14   ;
00012 F80F 86   51                      LDA     #$51    ;configure ACIA
00013 F811 B7   EC14                    STA     $EC14   ; for 7 bits, no parity
00014 F814 86   00                      LDA     #$00    ;zero all RAM
00015 F816 1C   FB                      ANDCC   #$FB    ; (locations
00016 F818 8E   8000                    LDX     #$8.    ; $8000-$DFFF)
00017 F81B A7   80      AGAIN           STA     ,X+
00018 F81D 8C   DFFF                    CMPX    #$DFFF
00019 F820 26   F9                      BNE     AGAIN
00020 F822 86   31                      LDA     #$30
00021 F824 B7   E002                    STA     $E003   ;select Bank A (AMET)
00022 F827 86   7E                      LDA     #$7E    ;set up "pseudo-interrupt
00023 F829 B7   DFFD                    STA     $DFFD   ; vector" jump
                                                          instruction
00024 F82C 0C   FB                      ANDCC   #$FB
00025 F82E 7E   0050                    JMP     $0050   ;start AMET
00026 F831 0C   FB                      ANDCC   #$FB
00027 F833 86   38                      LDA     #$E8
00028 F835 B7   E003                    STA     $E003   ;select Bank B (BMET)
00029 F838 8E   A000                    LDX     #$A000  ;zero RAM
00030 F83B 86   00                      LDA     #$00    ; from $A000 to $DFFF
00031 F83D A7   80      BOZO            STA     ,X+
00032 F83F 8C   E000                    CMPX    #$E000
00033 F842 26   F9                      BNE     BOZO
00034 F844 1C   FB                      ANDCC   #$FB
00035 F846 86   7E                      LDA     #$7E    ;cause IRQ interrupts
00036 F848 B7   DFFD                    STA     $DFFD   ;to trigger
00037 F84B CC   F900                    LDD     #$F900  ;System Error Message
00038 F84E FD   DFFE                    STD     $DFFE
00039 F851 7E   0150                    JMP     $0150   ;start BMET
00040 F900                              ORG     $F900   ;SYSTEM ERROR MESSAGE
                                                          ROUTINE
00041 F900 1C   FB                      ANDCC   #$FB
00042 F902 8E   FA00                    LDX     #$FA00  ;X reg = loc. of 1st
                                                          char.
00043 F905 E6   80      DOG     LDB     ,X+     ;get a character, incr. X
00044 F907 B6   EC14    PRINT1  LDA     $EC14   ;loop until terminal
00045 F90A 84   92                      ANDA    #$02    ; is free.
00046 F90C 27   F9                      BEQ     PRINT1
00047 F90E F7   EC15                    STB     $EC15   ;send char. to terminal
```

```
00048 F911 1C    FB              ANDCC   #$FB
00049 F913 8C    FA37            CMPX    #$FA37   ;reached end of message?
00050 F916 27    03              BEQ     BABOON   ;if so, go to return
                                                   (BABOON)
00051 F918 7E    F905            JMP     DOG      ;if not, get next
                                                   char.(DOG)
00052 F91B 12            BABOON  NOP
00053 F91C 3B                    RTI              ;return from interrupt
00054 FA00                       ORF     $FA00
00055 FA00       0D0A            FDB     $0D0A,$0A0A
00056 FA04       2A              FCB     $2A      ; *
00057 FA05       2A              FCB     $2A      ; *
00058 FA06       2A              FCB     $2A      ; *
00059 FA07       2A              FCB     $2A      ; *
00060 FA08       20              FCB     $20      ;
00061 FA09       57              FCB     $57      ; W
00062 FA0A       41              FCB     $41      ; A
00063 FA0B       52              FCB     $52      ; R
00064 FA0C       4E              FCB     $4E      ; N
00065 FA0D       49              FCB     $49      ; I
00066 FA0E       4E              FCB     $4E      ; N
00067 FA0F       47              FCB     $47      ; G
00068 FA10       20              FCB     $20      ;
00069 FA11       53              FCB     $53      ; S
00070 FA12       59              FCB     $59      ; Y
00071 FA13       5354            FDB     $5354    ; ST
00072 FA15       454D            FDB     $454D    ; EM
00073 FA17       2049            FDB     $2049    ;  I
00074 FA19       4E54            FDB     $4E54    ; NT
00075 FA1B       4552            FDB     $4552    ; ER
00076 FA1D       5255            FDB     $5255    ; RU
00077 FA1F       5054            FDB     $5054    ; PT
00078 FA21       2045            FDB     $2045    ;  E
00079 FA23       5252            FDB     $5252    ; RR
00080 FA25       4F52            FDB     $4F52    ; OR
00081 FA27       204F            FDB     $204F    ;  H
00082 FA29       4153            FDB     $4153    ; AS
00083 FA2B       204F            FDB     $204F    ;  O
00084 FA2D       4343            FDB     $4343    ; CC
00085 FA2F       5552            FDB     $5552    ; UR
00086 FA31       4544            FDB     $4544    ; ED
00087 FA33       202A            FDB     $202A    ;  *
00088 FA35       2A2A            FDB     $2A2A    ; **
00089 FA37       2A2A            FDB     $2A2A    ; **
00090 FFF2                       ORG     $FFF2    ;interrupt vectors:
00091 FFF2       F900            FDB     $F900    ;SW3, to System Error Msg.
00092 FFF4       F900            FDB     $F900    ;SW2, to System Error Msg.
00093 FFF6       F900            FDB     $F900    ;FIRQ,to System Error Msg.
00094 FFF8       DFFD            FDB     $DFFD    ;IRQ,to pseudo-interrupt
                                                   vector
00095 FFFA       F900            FDB     $F900    ;SWI, to System Error Msg.
00096 FFFC       F900            FDB     $F900    ;NMI, to System Error Msg.
00097 FFFE       F800            FDB     $F800    ;RESET, to Monitor
```

```
PAGE 001  ADVANCE .SA:1            UMET-1                    15 DEC 1982

0001        SUBROUTINE ADVANC (JJ,TSTOP,TLANCH,TGMDAQ,MASK)
0002 *
0003 ********************************* COMMON BLOCK ***********************************
0004 *
0005             INCLUDE 'COMMON.SA'
0006 *
0007 ***************************************************************************************
0008 C
0009 C           ADVANCE, 5 NEW RAW DATA POINTS.
0010 *
0011 C
0012          MASK=0
0013 C
0014        DO  2 JI=1,5
0015        J5= JI+5
0016         TIME(JI)=TIME(J5)
0017         FREQ(JI)=FREQ(J5)
0018         AZ(JI)=AZ(J5)
0019         EL(JI)=EL(J5)
0020      2 CONTINUE
0021 *
0022 ****     HANDLE REAL TIME DATA ***********************************************
0023 4887   DO 3 J=6,10
0024 3223   IF(IFLAG .GT. 0)GOTO 3224
0025        CALL STOPER
0026        IF(ISTPER .NE. 1)GOTO 3223 ;IF STOP IS HIT ISTPER=1
0027        MASK=4
0028        RETURN
0029 3224   INTMSK=1
0030        IF(IBOT .GE. 150)IBOT=0   ;IBOT POINTS TO OLDEST NOT YET PROCESSED POI
0031        IDUM=IBOT*10
0032        IBOT=IBOT+1               ;THE IDATA ARRAY IS ONLY 75 POINTS LONG
0033        IFLAG=IFLAG-1
0034        DUMMY=IDATA(IDUM+9)
0035        DUMMY1=IDATA(IDUM+10)
0036        FREQ(J)=(DUMMY*1000)+DUMMY1
0037        DUMMY=IDATA(IDUM+6)
0038        DUMMY1=IDATA(IDUM+5)
0039        EL(J)=DUMMY1+(DUMMY*.01)
0040        DUMMY=IDATA(IDUM+8)
0041        DUMMY1=IDATA(IDUM+7)
0042        AZ(J)=DUMMY1+(DUMMY*.01)
0043        DUMMY=IDATA(IDUM+4)
0044        DUMMY1=IDATA(IDUM+3)
0045        SECS=DUMMY1+(DUMMY*.1)
0046        DUMMY=IDATA(IDUM+2)
0047        AMINU=DUMMY*60.+SECS
0048        HOURS=(IDATA(IDUM+1))*3600.
0049        TIME(J)=HOURS+AMINU-TLANCH  ;TIME IS IN SECONDS FROM LAUNCH
0050        IF(INTMSK .EQ. 1)GOTO 444
0051        CALL GETBCK
0052 444    INTMSK=0
0053        IF(TIME(J) .LT. TIME(J-1))GOTO 5
0054        DUMMY=TIME(J-1)+60.
0055        IF(TIME(J) .GT. DUMMY)GOTO 5
0056 *
0057 ********** REAL TIME DATA NOW IN PROGRAM **************
0058 2204   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)*.000001)       ;DATA COMES IN IN USEC
0064 *
0065         IF(FREQ(J) .LT. 4.8 .OR. FREQ(J) .GT. 200.)GO TO 5
0066         GO TO 3
0067 5       J=J-1
0068  3       CONTINUE
0069          IF(TIME( 9) .GT. TSTOP) GOTO 100
0070 *
0071 *                  INCREMENT TABLE AND ENTER GMD ANGLES
0072 *
0073 51       IF(TIME(4) .LT. VL(1,LIST)) GO TO 53
0074          IF(TIME(4) .LT. TGMDAQ) GO TO 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
```

```
0001         SUBROUTINE ANGLE
0002 *
0003 ************************************COMMON BLOCK*****************
0004 *
0005         INCLUDE 'COMMON.SA'
0006 *
0007 ************************************************************************
0008         DIMENSION NIAZ(10),NIEL(10),IAZ(10),IEL(10)
0009 *
0010 *                INITIALIZE AND QUANTIZE
0011 *
0012         DO 11 K=1,10
0013         NIAZ(K)=0
0014         NIEL(K)=0
0015         IAZ(K)=AZ(K)+0.5
0016         IEL(K)=EL(K)+0.5
0017   11    CONTINUE
0018 *
0019 *                COUNT FOR DISTRIBUTION
0020 *
0021         DO 10 K=1,9
0022          LL=11-K
0023         DO 12 L=1,LL
0024         IF(IAZ(LL) .EQ. IAZ(L)) NIAZ(LL)=NIAZ(LL)+1
0025        IF(IEL(LL) .EQ. IEL(L)) NIEL(LL)=NIEL(LL)+1
0026   12   CONTINUE
0027   10   CONTINUE
0028 *
0029 *                DETERMINE MODE INTERVAL
0030 *
0031         KMA=1
0032         KME=1
0033         DO 13 K=2,10
0034         IF(NIAZ(K) .GT. NIAZ(KMA)) KMA=K
0035         IF(NIEL(K) .GT. NIEL(KME)) KME=K
0036   13    CONTINUE
0037 *
0038 *                COMPUTE MEAN IN MODE INTERVAL
0039 *
0040         SUMA=0.
0041         NSUMA=0
0042         DO 14 K=1,10
0043         IDUM=IAZ(K)-IAZ(KMA)
0044         IDUM1=ABS(IDUM)
0045         IF(IDUM1 .GT. 2) GOTO 14
0046         SUMA=SUMA+AZ(K)
0047         NSUMA=NSUMA+1
0048    14   CONTINUE
0049         SUME=0.
0050         NSUME=0
0051         DO 15 K=1,10
0052         IF(IEL(K) .NE. IEL(KME)) GO TO 15
0053         SUME=SUME+EL(K)
0054         NSUME=NSUME+1
0055    15   CONTINUE
0056 *
0057 *                OUTPUT
0058 *
```

```
0059          DUMMY = NSUMA          ;THIS CHANGES AN INTEGER DATA TO FLOATING POIN
0060          DUMMY1= NSUME
0061          AZ(5)=SUMA/DUMMY
0062          EL(5)=SUME/DUMMY1
0063          RETURN
0064          END
```

```
      COMMON IFLAG,JUNK(7),ISTPER,IXXX,INTMSK,IRAM(10),ITOP,IBOT ;FOR REAL TIME
      COMMON IDATA(1500),VL(7,120),SECS,DUMMY,DUMMY1,IDUMM,IDUMM1,AMINU
      COMMON IDSTAT,IDSOND,IMONTH,IDAY,IYEAR,HEIGHT,FF0,FTEMP0
      COMMON FR0,RECTF,CALRH,ICRBN,CBRST,NCDS,IOUT,VSFC,DSFC   ;PASS TO FRD
      COMMON SURT,SURH,IAZU,I1,I2,TS3       ;PASS DATA TO FRD
      COMMON LIST,DLIST,SIGMAX,SIGMIN,HGATE,IN,SIGLEV,NSUM,FSUM
      COMMON TBRST,IOIN,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)
```

```
0001 C                SUBROUTINE     DECOM    DECOM     DECOM    DECOM
0002 C
0003       SUBROUTINE DECOM(JK,TNOH,DSL,FHUM,ICR1,RFL,PFL,IDG)
0004 *
0005 ************************ COMMON BLOCK**************************************
0006 *
0007        INCLUDE 'COMMON.SA'
0008 *
0009 ********************************************************************************
0010 *
0011        DIMENSION DWELT(10), TDWELT(10), SDWELT(10)
0012 *
0013 C
0014 C
0015 C                   INITIALIZE DECOM
0016 C
0017        IF(JK .GT. 1)GOTO 5
0018        DRPFL=2.0
0019        RFSUM=0.0
0020        RTSUM=0.0
0021        TR=0.0
0022        AMLT=100.0
0023        MLT=100
0024        GTEMP=4.0
0025        TSL=0.0
0026        ESL=0.0
0027        TR1=0.0
0028        INCH=0
0029        JKR=0
0030        NOH=0
0031        JKP=0
0032        ICM=0
0033        JKR1=0
0034        M1=1
0035        ICR=0
0036        KROSS=0
0037        NXTP=0
0038 *
0039 *
0040 *
0041 C               CONTACT NRS., ICR, ETC., ARE ALL MULTIPLIED BY MLT
0042 C               IN SUBROUTINE DECOM ONLY.
0043 C             ICOND(2, ) IS NOT MULTIPLIED BY MLT. ALSO IN UMET, THE
0044 C              OLD VALUES REVISED IN BACK ASSIGNING (USING
0045 C              1000X SHIFT, ETC.)ARE DELETED BECAUSE INTEGERS ARE
0046 C              LIMITED TO 2**15 IE 32K.
0047 *
0048  5    CONTINUE
0049       IF(NXTP .EQ. 0)NXTP = 30*MLT
0050 *
0051 C                   REJECT SHORT DWELLS
0052 *
0053       IF(COND(2,JK) .GT. 1.9)GO TO 8
0054       ICOND(1,JK) = 8
0055       GO TO 900
0056 C
0057  8     T = COND(1,JK)                ;ELAPSED TIME TO LEADING EDGE
0058        DWELL = COND(2,JK)            ;DWELL TIME
```

```
0059          D = COND(3,JK)                ;VALUE (FREQ)
0060         TB = T + DWELL/2.
0061         IF(D .GT. RFL)GO TO 200
0062 *
0063 *
0064 C                        PROCESS  -A-
0065 *
0066 *
0067         IF(TBRST .GT. 1.E20)GO TO 10
0068 *
0069  178    IF(M1 .LT. 0)GO TO 179
0070         ICR1 = ICR
0071         SLOP1 = SLOPE
0072
0073  179    JKR1 = JKR
0074         TR1 = TR
0075         M1 = 1
0076         KROSS = 0
0077         GO TO 78
0078 *
0079 10      IF(INCH .NE. 2)GO TO 78       ;10
0080         IF(JKR .EQ. 0)GO TO 782
0081         GO TO 600
0082 *
0083 C                   FORWARD-ASSIGN BETWEEN REFERENCES
0084 *
0085 *
0086 C                   TEMPERATURE GATE
0087 *
0088  78     CONTINUE
0089         TF = DSL + ESL *(TB - TSL)
0090         DUMMY =D-TF
0091         IF (DUMMY .GT. 0.0) GOTO 2200
0092         DUMMY = DUMMY*(-1)
0093 2200    IF(DUMMY .GT. GTEMP)GO TO 783
0094
0095 C                   A TEMPERATURE DATUM
0096 *
0097  782    ESL1 = ESL
0098         ESL = .8*ESL + .2*(D-DSL)/(TB-TSL)
0099         DUMMY = ESL-ESL1
0100         IF (DUMMY .GT.0.0) GOTO 2210
0101         DUMMY = DUMMY*(-1)
0102 2210    IF(DUMMY .GT. .2)ESL = ESL1
0103  781    DSL = .8*DSL + .2*D
0104         TSL = TB
0105         ICOND(1,JK) = 1
0106         INCH = 1
0107         GO TO 900
0108 *
0109 783     IF(NOH .EQ. 1)GO TO 784    ;783
0110         IF(JKR .EQ. 0)GO TO 785
0111 *
0112 C                      ASSURE DWELL SUCCEEDING THE REFERENCE IS A TEMPERATURE.
0113 *
0114         IF(T .GT. TR + .8*SLOPE)GO TO 785
0115 *
0116 C                   RESTORE TEMPERATURE FREQUENCY GATE
```

```
0117 *
0118  7831 ESL = 0.0
0119       KROSS = 1
0120       GO TO 781
0121
0122  784  IF(INCH .EQ. 2)GO TO 7831
0123       ICOND(1,JK) = 7
0124       INCH = 5
0125       GO TO 900
0126
0127 C                  PROCESS THIS NON-REFERENCE, NON-TEMPERATURE
0128 *
0129  785  CONTINUE
0130       IF(INCH .NE. 1) GO TO 788
0131 *
0132 C                  A CONTACT SWITCH POINT
0133 *
0134       IF(ICM .NE. 0) GO TO 786
0135 *
0136 C                  FIRST CONTACT SWITCH POINT
0137 *
0138       IDUMMY=ICR1/MLT
0139       ICM = (IDUMMY+1)*MLT
0140       IDUMMY = ICM-ICR1
0141       DUMMY = IDUMMY
0142       SLOP2 = AMLT*T/DUMMY
0143        GO TO 787
0144
0145  786  IDC = (T-T2)/SLOP2 + .5
0146       IDC = IDC*MLT
0147       IF(IDC .LT. MLT)GO TO 789
0148       IDUMMY = ICR1/(5*MLT)
0149       IDUMM1 = IDUMMY*5+5
0150       IF(ICM+IDC .LT.IDUMM1*MLT)GO TO 7861
0151       ICOND(1,JK) = 6
0152       KROSS = 1
0153       INCH = 5
0154       GO TO 900
0155 *
0156  7861 CONTINUE
0157       ICM = ICM + IDC
0158       SLOP2 = AMLT*(T-T2)/IDC
0159 787   ICOND(2,JK) = ICM/MLT   ;787
0160       T2 = T
0161 C                  AN HUMIDITY DATUM
0162 *
0163  788  ICOND(1,JK) = 4
0164       INCH = 4
0165       FHUM = D
0166       GO TO 900
0167
0168 C                  EARLY CONTACT, POSSIBLE BALLOON DIP
0169
0170  789  M1 = 0
0171       ICOND(1,JK) = 5
0172       INCH = 5
0173       GO TO 900
0174 *
```

```
0175 C                          PROCESS -B-
0176 *
0177 C                  PROCESS THE PRECEDING REFERENCE
0178 C                          CHECK FOR BURST AND BACK-ASSIGN AS APPROPRIATE.
0179 *
0180  600  COND(3,JKR) = RFSUM/RTSUM
0181       COND(2,JKR) = RTSUM
0182       TR = COND(1,JKR)
0183       IF(ICR .NE. 0)GO TO 610
0184 C                FIRST REFERENCE
0185 *
0186        ICRB = TR/90. + .5
0187        ICRB1 = ICR1/(5*MLT)
0188        IF(ICRB .EQ. 0)ICRB=1
0189       ICR=(ICRB1+ICRB)*5*MLT
0190       IDUMMY=ICR-ICR1
0191       DUMMY = IDUMMY
0192       SLOPE = AMLT*TR/DUMMY
0193       ICR1 = ICR
0194       SLOP1 = 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                          ;CHANGES INTEGER TO REAL #
0207       SLOPE = AMLT*(TR-TR1)/DUMMY
0208       M = SLOPE/SLOP1 + 0.5
0209       IF(M .GT. 1)GO TO 620
0210       IF(M .EQ. 1)GO TO 644
0211 C                 M = 0  INTERPRETED AS BALLOON DIP, BAROSWITCH REVERSAL.
0212 C                 SUCH AN EARLY REFERENCE IS IGNORED FOR PRESSURE
0213
0214       IDUMMY = ICR/MLT
0215       DUMMY = TR/60
0216       WRITE(IOUT,944)IDUMMY,DUMMY
0217  944   FORMAT(1X,'CHECK FOR BALLOON DIP AFTER CONTACT ',I3,
0218 &        ' NEAR',F5.1,' MINUTES.')
0219       INCH = 1
0220       M1 = -1
0221       ICR = ICR1
0222       SLOPE = SLOP1
0223       GO TO 180
0224
0225  644   CONTINUE
0226       DUMMY = SLOPE/SLOP1-1
0227       IF (DUMMY .GT. 0.0) GOTO 2220  ;TAKE ABSOLUTE VALUE OF DUMMY
0228       DUMMY = DUMMY*(-1)
0229 2220  IF(MN .LT. 1 .AND. DUMMY .LT. .3)MN=-1
0230       IF(MN .NE. -1 .OR. M1 .NE. 0)GO TO 645
0231       M1 = 2
0232       DUMMY = TR1/60.0
```

```
0233        DUMMY1 = TR/60.0
0234        WRITE(IOUT,6450) DUMMY,DUMMY1
0235  6450  FORMAT(' EARLY CONTACT FOUND IS FALSE BECAUSE NO',
0236 &        ' BALLOON DIF BETWEEN ',F6.2,' AND ',F6.2,' MINUTES.'
0237 &       ,'  WILL BACK-ASSIGN!')
0238  645   ICOND(2,JKR) = ICR/MLT      ;645
0239        ICM = ICR
0240        GTSW = 0.1 * SLOPE
0241        HUMGT = 0.25 * SLOPE
0242 C
0243 C                        SET 'NO HUMIDITY' FLAG ABOVE CONTACT NUMBER 135
0244 *
0245        IF(NOH .EQ. 1)GO TO 50
0246         IF (ICR .LT. 135*MLT) GO TO 120
0247         NOH=1
0248         TNOH = TR
0249        GTEMP = 6.
0250 *
0251 C                  ADVANCE DWELT ARRAY,AND TEST FOR BURST
0252 *
0253     50  IF (TR .LT. 3000.) GO TO 55
0254        DO 52  IS = 1,9
0255         IDUMMY = 11-IS
0256         IDUMM1= 10-IS
0257         DWELT(IDUMMY) =  DWELT(IDUMM1)
0258         SDWELT(IDUMMY) = SDWELT(IDUMM1)
0259     52 TDWELT(IDUMMY) = TDWELT(IDUMM1)
0260        DWELT(1) = SLOPE
0261         SDWELT(1) = 0.0
0262         DO 51 ISS = 1,4
0263     51  SDWELT(1) = SDWELT(1) + DWELT(ISS)
0264         SDWELT(1) = SDWELT(1) / 4.0
0265        TDWELT(1) = TR
0266 C      WRITE(IOUT,951) TDWELT
0267 C      WRITE(IOUT,951)  DWELT
0268 C       WRITE(IOUT,951) SDWELT
0269 C 951 FORMAT( 1X,10F12.1)
0270        IDUMMY = CBRST                              ;TRUNCATES IE XX.X=XX
0271        IF (ICR .LT. IDUMMY*MLT)GO TO 54
0272        DUMMY=IDUMMY                                ;BACK TO REAL (XX.0)
0273        TBRST = TR + (CBRST-DUMMY)*SLOPE
0274        GO TO 56
0275  54      CONTINUE
0276         IBRST = 0
0277        DO 53  IS = 1,3
0278     53  IF(DWELT(IS) .GT. 15.0 .AND. SDWELT(IS) .LT. SDWELT(IS+1) .AND.
0279 &              SDWELT(IS+3) .GT. 70.0) IBRST = IBRST + 1
0280         IF(IBRST .LT. 3 .OR. TR .LT. 4000.) GO TO 55
0281 *
0282 C                        BURST CONDITIONS ENCOUNTERED
0283 *
0284         TBRST = TDWELT(4)
0285  56    TBM = TBRST / 60.
0286        WRITE(IOUT,950) TBM,IBRST
0287   950 FORMAT(1X,'BURST AT', F6.1, ' MINUTES.  IBRST =', I3)
0288  55     CONTINUE
0289  17    IF(COND(2,JKR) .GT. PFL)GO TO 130
0290 *
```

```
0291 C                  ----- ADJUST REFERENCE THRESHHOLDS
0292 *
0293        WRITE(IOUT,8675)COND(3,JKR),RFL,PFL
0294 8675  FORMAT('LOND3',F8.3,' RFL',F8.3,' PFL',F8.3)
0295        RFL = 0.6 * RFL + 0.4 * (COND(3,JKR)-10.0)
0296        PFL = 0.6 * PFL + 0.4 * (RFL+DRPFL+10.0)
0297        GO TO 160
0298 C                  PROCESS THIS HIGH REFERENCE
0299 *
0300  130   DRPFL = 0.6 * DRPFL + 0.4 * (COND(3,JKR)-RFL-10.0)*.5
0301        ICOND(1,JKR) = 3
0302
0303        IF(JKP .EQ. 0)GO TO 131 .
0304 C
0305 C            CORRECT NXTP WHEN APPARENT THAT A HIGH REF. (P) PT. WAS MISSED
0306 C
0307        TP12 = 15.*SLOPE
0308         IF (NXTP .GT. 135*MLT) TP12 = 5.*SLOPE
0309        IDUMMY=15*MLT
0310        IF(NXTP.LT.135*MLT.AND.(T-TP1).GT.(1.5*TP12).AND.
0311 &      MOD(ICR,IDUMMY).EQ. 0 .AND.ICR.LE.135*MLT)NXTP=ICR
0312         DUMMY=T-TP1-2.*TP12
0313         IF (DUMMY .GT. 0.0) GOTO 2230
0314         DUMMY =DUMMY*(-1)
0315  2230   IF (NXTP .LT. 135*MLT .AND. DUMMY .LT. .5*TP12 .AND.
0316 &      ICR .NE. NXTP) NXTP = NXTP + 15*MLT
0317         IDUMMY = 5*MLT
0318         IF (NXTP .GT. 135*MLT .AND. (T-TP1) .GT. (1.8*TP12) .AND. MOD(
0319 &      ICR,IDUMMY) .EQ. 0 .AND. ICR .GT. NXTP) NXTP = ICR
0320         IF (NXTP .GT. 135*MLT .AND. DUMMY .LT. .5*TP12
0321 &      .AND. ICR .NE. NXTP) NXTP = NXTP + 5*MLT
0322 *
0323  131   IF(ICR .EQ. NXTP)GO TO 132
0324        WRITE(IOUT,1131)ICR,NXTP
0325  1131  FORMAT(' CONTACT NUMBER ',I8,' NOT EQUAL TO NXTP ',I8)
0326
0327  132   NXTP = NXTP + 5*MLT
0328        IF  (ICR.LT.135*MLT)  NXTP = NXTP + 10*MLT
0329        TP1 = TR
0330        JKP = JKR
0331 C               BACK-ASSIGN
0332 C          DECOMMUTATE TEMPERATURE AND HUMIDITY UP TO THE CURRENT REFERENCE.
0333
0334 C              ASSURE DWELL PRECEDING THIS REFERENCE IS A TEMPERATURE
0335 *
0336  180   CONTINUE
0337        IF(NOH .EQ. 1)GO TO 178
0338        IF(COND(1,JKR-1) .LT. TR - .8*SLOPE)GO TO 181
0339        IF(ICOND(1,JKR-1) .EQ. 1)GO TO 181
0340        ICOND(1,JKR-1) = 10 + ICOND(1,JKR-1)
0341        ESL = 0.
0342        TSL = COND(1,JKR-1) + COND(2,JKR-1)/2.
0343        DSL = COND(3,JKR-1)
0344        KROSS = 1
0345  181   IF(KROSS .NE. 1)GO TO 188
0346        JNFIN = JKR-2
0347        INCHN = 2
0348        ESLN = ESL
```

```
0349        TSLN = TSL
0350        DSLN = DSL
0351        IDUM2=JNFIN-JNSTRT+1
0352        DO 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        TBN = TN + DWELLN/2.
0365        TS = (TN-TR1)/SLOPE
0366        NCT = TS + .5
0367        DUMMY = NCT
0368        T0 = DUMMY*SLOPE + TR1
0369        TFN = DSLN + ESLN * (TBN-TSLN)
0370        DUMMY = DN-TFN
0371        IF (DUMMY .GT. 0.0) GOTO 2240
0372        DUMMY = DUMMY*(-1)
0373 2240  IF(NCT.NE.0.AND.NCT.NE.1.AND.DUMMY.GT.GTEMP)
0374 &     GOTO 183
0375        IF(TBN .GE. (T0-GTSW))GO TO 183
0376 *
0377 C                   A TEMPERATURE DATUM
0378 *
0379 1821  ICOND(1,JN) = 10 + ICOND(1,JN)
0380        ESLN1 = ESLN
0381        ESLN = 0.8*ESLN + 0.2*(DN - DSLN)/(TBN - TSLN)
0382        DUMMY = ESLN-ESLN1
0383        IF (DUMMY .GT. 0.0) GOTO 2250
0384        DUMMY = DUMMY*(-1)
0385 2250  IF(DUMMY .GT. .2) ESLN = ESLN1
0386        DSLN = .8*DSLN + .2*DN
0387        TSLN = TBN
0388        INCHN = 1
0389        GO TO 1874
0390 C              BACK-ASSIGN CHANNEL AND CONTACT NUMBERS
0391 *
0392 183   DUMMY = TN-T0
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(DUMMY.GE. GTSW .OR. DUMMY1 .LE.GTEMP)GOTO 1873
0399        IF(NCT .GE. 5 .OR. NCT .LE. 0)GO TO 1873
0400        IF(ICOND(2,JN+1)/1.EQ.ICOND(2,JKR1)+NCT)GO TO 1873
0401        ICOND(2,JN) = (ICOND(2,JKR1) + 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  1870 CONTINUE
0414 *
0415 C                    WRAP-UP PROCESS -B-
0416 *
0417  188   IF(M1 .LT. 0) GO TO 189
0418        ICR1 = ICR
0419        SLOP1 = SLOPE
0420
0421  189   JKR1 = JKR
0422        TR1 = TR
0423        T2 = TR
0424        SLOP2 = SLOPE
0425        M1 = 1
0426        KROSS = 0
0427        JNSTRT = JK + 1
0428        GO TO 78
0429 *
0430 C                           STORE REFERENCE DATA
0431 *
0432    200  CONTINUE
0433        IF(INCH .EQ. 0) GO TO 220
0434         IF(INCH .EQ. 2/GO TO 220
0435 *
0436 C                   A CONTACT SWITCH POINT
0437 *
0438         JKR = JK
0439        RFSUM = 0.0
0440        RTSUM = 0.0
0441 *
0442 C                     CUMULATION OF REFERENCE MEAN FREQUENCY
0443 *
0444    220   RFSUM = RFSUM + D * DWELL
0445         RTSUM = RTSUM + DWELL
0446        ICOND(1,JK) = 2
0447         INCH = 2
0448 *
0449 900     IF(IDG .LE. 2)RETURN       ;900
0450         IF(MOD(ITCNT,50).EQ.0)WRITE(IOUT,9001)
0451         ITCNT = ITCNT + 1
0452         WRITE(IOUT,9000)(COND(I,JK),I=1,3),(ICOND(I,JK),I=1,2),JK,
0453 &        INCH, NOH, RFL, DRPFL, TF, FHUM, ESL, SLOPE,KROSS,M1
0454  9001 FORMAT('1DECOM OUTPUT: -----COND(I,JK),I=1,3',
0455 &        '-----ICOND   JK INCH  NOH    RFL',
0456 &        '  DRPFL     TF   FHUM    ESL  SLOPE KROSS    M1')
0457  9000 FORMAT(' DECOM OUTPUT: ',3F7.1,5I5,4F7.1,F7.4,F7.2,2I6)
0458
0459
0460         RETURN
0461         END
```

```
PAGE 001  GETBCK  .SA:1            UMET-1                    15 DEC 1982

0001  OPT L,REL
0002  OPT LLE=80
0003  NAM GETBCK
0004  IDNT GETBCK
0005  XDEF GETBCK
0006 *
0007 ************************************************************************
0008 *       THIS PROGRAM IS USED TO FORMAT THE DATA STORED IN BYTES 1-10
0009 *       DURING AN INTERRUPT.NORMAL OPERATION IS TO DO THIS FORMAT
0010 *       DURING THE INTERRUP, BUT IF WE ARE BRINGING DATA INTO THE MAIN
0011 *       PROGRAM WE DONT WANT TO START CHANGING THE DATA STACK OR POINTERS.
0012 *       TO SOLVE THE PROBLEM A FLAG IS SET THAT TELLS THE INTERRUPT NOT
0013 *       TO FORMAT THE DATA. THEN AFTER THE DATA IS IN THE FORTRAN
0014 *       PROGRAM IT SEES IF INTMSK (THE FLAG) HAS BEEN SET TO 2 BY
0015 *       AN INTERRUP. IF SO THIS PROGRAM IS CALLED.
0016 *
0017  CSCT
0018 IFLAG RMB 2
0019 BYTE10 RMB 1
0020 BYTE9 RMB 1
0021 BYTE8 RMB 1
0022 BYTE7 RMB 1
0023 BYTE6 RMB 1
0024 BYTE5 RMB 1
0025 BYTE4 RMB 1
0026 BYTE3 RMB 1
0027 BYTE2 RMB 1
0028 BYTE1 RMB 1
0029 BYTE11 RMB 1
0030 BYTE12 RMB 1
0031 BYTE13 RMB 1
0032 BYTE14 RMB 1
0033 BYTE15 RMB 1
0034 BYTE16 RMB 1
0035 TEMPX RMB 2
0036 INTMSK RMB 2
0037 IRAM1 RMB 2
0038 IRAM2 RMB 2
0039 IRAM3 RMB 2
0040 IRAM4 RMB 2
0041 IRAM5 RMB 2
0042 IRAM6 RMB 2
0043 IRAM7 RMB 2
0044 IRAM8 RMB 2
0045 IRAM9 RMB 2
0046 IRAM10 RMB 2
0047 ITOP RMB 2
0048 IBOT RMB 2
0049 IDATA RMB 1500     ;IRAM ARE STACKED INTO THE IDATA ARRY TO BE PROCESSED
0050 **********************
0051 *
0052 *
0053 *  PROGRAM FOLLOWS
0054 *
0055 *           RAW DATA IS NOW STORED FROM BYTE1-BYTE10
0056 *           THIS WAS DONE IN THE INTERUP.
0057 *
0058 *           FORMAT THE DATA INTO INTERGER IN IRAM USING BYTE11-BYTE16
```

```
0059 *          AS SCRATCH PAD. THEY ARE STORED AS FOLLOWS:
0060 *          IRAM1=HOURS
0061 *          IRAM2=MIN
0062 *          IRAM3=SEC
0063 *          IRAM4=.SEC
0064 *          IRAM5=DEGREES OF EL
0065 *          IRAM6=.DEGREES OF EL
0066 *          IRAM7=DEGREES OF AZ
0067 *          IRAM8=.DEGREES OF AZ
0068 *          IRAM9=THOUSANDS PLACES IN MET WORD
0069 *          IRAM10=LOWEST 3 VALUES OF MET WORD (IE UP TO 999)
0070 *
0071 *
0072  PSCT
0073 GETBCK PSHS X       ;SAVE THE FORTRAN X REGISTER
0074 *                    START BY FINDING VALUE OF HOURS
0075  LDB BYTE10
0076  LSRB
0077  LSRB
0078  STB BYTE11
0079  LSRB
0080  LSRB
0081  LSRB
0082  LSRB
0083  LDA BYTE11
0084  ANDA #$0F
0085  STA BYTE11
0086  LDA #$0A
0087  MUL
0088  ADDB BYTE11
0089  STD IRAM1
0090 *          NOW DO MIN
0091  LDA BYTE10
0092  ANDA #$03
0093  STA BYTE11
0094  LDA BYTE9
0095  ASLA
0096  BCC NOCAR
0097  LDA BYTE11
0098  LSLA
0099  ORA #$01
0100  STA BYTE12
0101  BRA LOWMIN
0102 NOCAR LDA BYTE11
0103  LSLA
0104  STA BYTE12
0105 LOWMIN LDA BYTE9
0106  ANDA #$7F
0107  LSRA
0108  LSRA
0109  LSRA
0110  STA BYTE13
0111  LDB BYTE12
0112  LDA #$0A
0113  MUL
0114  ADDB BYTE13
0115  STD IRAM2
0116 *          NOW DO SECONDS
```

```
0117  LDA BYTE8
0118  LSRA
0119  LSRA
0120  LSRA
0121  LSRA
0122  STA BYTE11
0123  LDB BYTE9
0124  ANDB #$07
0125  LDA #$0A
0126  MUL
0127  ADDB BYTE11
0128  STD IRAM3
0129 *          NOW DO TENTHS OF SECONDS
0130  LDB BYTE8
0131  ANDB #$0F
0132  LDA #$00
0133  STD IRAM4
0134 *
0135 *          TIME WORD DONE
0136 *          START ON EL WORD
0137  LDA BYTE7
0138  ANDA #$0F
0139  STA BYTE11
0140  LDB BYTE7
0141  ANDB #$F0
0142  LSRB
0143  LSRB
0144  LSRB
0145  LSRB
0146  LDA #$0A
0147  MUL
0148  ADDB BYTE11
0149  STD IRAM5
0150 *          DO PART OF DEGREE EL
0151  LDA BYTE6
0152  ANDA #$0F
0153  STA BYTE11
0154  LDB BYTE6
0155  ANDB #$F0
0156  LSRB
0157  LSRB
0158  LSRB
0159  LSRB
0160  LDA #$0A
0161  MUL
0162  ADDB BYTE11
0163  STD IRAM6
0164 *         DONE WITH EL
0165 *         START ON AZ WORD
0166  LDB BYTE5
0167  LSRB
0168  LSRB
0169  LSRB
0170  LSRB
0171  LDA #$64
0172  MUL
0173  STD BYTE13
0174  LDB BYTE5
```

```
0175  ANDB #$0F
0176  LDA #$0A
0177  MUL
0178  ADDD BYTE13
0179  STD BYTE13
0180  LDA #$00
0181  LDB BYTE4
0182  LSRB
0183  LSRB
0184  LSRB
0185  LSRB
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  ADDB BYTE11
0200  STD IRAM8
0201 *           AZ WORD DONE
0702 *           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  ADDB 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  LSRB
0226  LSRB
0227  LSRB
0228  LSRB
0229  LDA #$0A
0230  MUL
0231  ADDB BYTE11
0232  ADDD BYTE13
```

```
0233  STD IRAM10
0234 *
0235 *            DATA IS FORMATTED AS INTERGERS
0236 *            NOW FORMATT INTO IDATA ARRY FOR USE IN MAIN
0237 *            MET PROGRAM.
0238 *
0239  LDD ITOP
0240  ADDD #$01
0241  CMPD #150
0242  BNE ISTOP
0243  LDD #$0000
0244 ISTOP STD ITOP
0245  CMPD IBOT
0246  BNE SETFLG
0247  LDD IBOT
0248  ADDD #$01
0249  CMPD #150
0250  BEQ ATTOP
0251  STD IBOT
0252  BRA SETFLG
0253 ATTOP LDD #$0000
0254  STD IBOT
0255 SETFLG LDD ITOP
0256  CMPD IBOT
0257  BLO WORKUP
0258  SUBD IBOT
0259  STD IFLAG
0260  BRA STOR
0261 WORKUP ADDD #150
0262  SUBD IBOT
0263  STD IFLAG
0264 STOR LDX TEMPX
0265  LDD IRAM1
0266  STD ,X++
0267  LDD IRAM2
0268  STD ,X++
0269  LDD IRAM3
0270  STD ,X++
0271  LDD IRAM4
0272  STD ,X++
0273  LDD IRAM5
0274  STD ,X++
0275  LDD IRAM6
0276  STD ,X++
0277  LDD IRAM7
0278  STD ,X++
0279  LDD IRAM8
0280  STD ,X++
0281  LDD IRAM9
0282  STD ,X++
0283  LDD IRAM10
0284  STD ,X++
0285  CMPX #$8CE6      ;HIGHEST ADDRESS OF DATA STACK
0286  BHS LOOPX
0287  STX TEMPX
0288  BRA DONE
0289 LOOPX LDX #$812E ;BOTTOM ADDRESS OF DATA STACK
0290  STX TEMPX
```

```
0291  LDD #$00
0292  STD ITOP
0293 DONE PULS X          ;RESTORES FORTRAN X REGISTER
0294  RTS
0295  END
```

```
0001 *INTERP  SUBROUTINE INTERP
0002 *
0003       SUBROUTINE INTERP(JK,TNOH,ISTOP,LCNTK)
0004 *
0005 ********************************* COMMON BLOCK ***********************************
0006 *
0007       INCLUDE 'COMMON.SA'
0008 *
0009 **********************************************************************************
0010 C
0011       DIMENSION T1(7),  T2(7) ,V1(7),ALOSS(7)
0012 *
0013 C          TOLERABLE TIME INTERVALS BETWEEN SIGNAL DATA
0014 C                   ( P, R, T, H )
0015 *
0016       ALOSS(1)=0.0
0017       ALOSS(2)=0.0
0018       ALOSS(3)=0.0
0019       ALOSS(4)=200.0
0020       ALOSS(5)=600.0
0021       ALOSS(6)=100.0
0022       ALOSS(7)=100.0
0023 *
0024       KNTCT = 0
0025       I4 = 1
0026       I5 = 1
0027       I6 = 1
0028       I7 = 1
0029       DO 1   I = 4,7
0030       V1(I) = 0.
0031     1 T2(I) =-0.1
0032        TLPCAL = 1.0E10
0033 *
0034 *
0035 C                DO EACH ROW (TIME) OF OUTPUT TABLE
0036 C
0037       DO 30 L = 1,LIST
0038        IF(VL(1,L) .GT. TBRST) GO TO 47
0039        IF(VL(1,L) .GT. TLPCAL) GO TO 42
0040        IF(VL(1,L) .GT. COND(1,JK)) GO TO 48
0041 C
0042 C               DO EACH COLUMN ENTRY (VARIABLE) OF THE OUTPUT TABLE
0043 C
0044 C               IF TL IS BRACKETED, INTERPOLATE
0045       DO 20 IV = 4,7
0046   10  IF( VL(1,L) .LE. T2(IV) ) GO TO 101
0047 *
0048 C          ADVANCE BRACKET BEFORE INTERPOLATING
0049       IJ = IV - 3
0050       GO TO ( 11, 12, 13, 14 ) , IJ
0051 *
0052 *               NEXT PRESSURE PAIRS
0053 *
0054    11 DO 111     I = I4,900
0055        IF(I .GT. JK .OR. COND(1,I) .GT. TBRST) GO TO 101
0056       IF (ICOND(2,I) .GT. KNTCT .AND. ICOND(1,I) .LT. 5) GO TO 112
0057        IF(ICOND(2,I) .GE. LCNTK) GO TO 100
0058  111    CONTINUE
```

```
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 *               NEXT REFERENCE FREQUENCY PAIRS
0069 *
0070    12 DO 121   I = I5,900
0071        IF(I .GT. JK .OR. COND(1,I) .GT. TBRST) 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 *               NEXT TEMPERATURE PAIRS
0084 *
0085    13 DO 131   I = I6,900
0086        IF(I .GT. JK .OR. COND(1,I) .GT. TBRST) 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   133  CONTINUE
0096        V1(6) = VL(6,L-2)
0097        T1(6) = VL(1,L-2)
0098        V2(6) = VL(6,L-1)
0099        T2(6) = VL(1,L-1)
0100        GO TO 10
0101 *
0102 *               NEXT RELATIVE HUMIDITY PAIR
0103 *
0104   14   IF(VL(1,L.  .GT.  TNOH) GO TO 20
0105        DO 141   I = I7,900
0106        IF(I .GT. JK .OR. COND(1,I) .GT. TNOH .OR. COND(1,I) .GT. TBRST)
0107 &              GO TO 101
0108        IF( ICOND(1,I) .EQ. 4 .AND. COND(3,I) .GT. 0.001) GO TO 142
0109  141   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 *
```

```
0117   100  TLPCAL = COND(1,I)
0118 *
0119 *                INTERPOLATE / EXTRAPOLATE
0120 *
0121   101  IF(IV .EQ. 7 .AND. VL(1,L) .GT. TNOH) GO TO 20
0122        IF( V1(IV).GT. .0001 .AND. (T2(IV) - T1(IV)) .GT. .0001) GO TO 190
0123        VL(IV,L)= V2(IV)
0124        GO TO 20
0125   190  VL(IV,L)=V1(IV)+((V2(IV)-V1(IV))/(T2(IV)-T1(IV)))*(VL(1,L)-T1(IV))
0126 *
0127 C            OUTPUT ZEROS FOR NO LOCAL SIGNAL
0128 *
0129        DUMMY = VL(1,L)-T1(IV) .
0130        DUMMY1 = VL(1,L)-T2(IV)
0131        IF (DUMMY .GT. 0) GOTO 2210
0132        DUMMY = DUMMY*(-1)
0133 2210   IF (DUMMY1 .GT. 0) GOTO 2200
0134        DUMMY1 = DUMMY1*(-1)
0135 2200   IF(DUMMY .GT. ALOSS(IV) .AND.
0136 &          DUMMY1 .GT. ALOSS(IV) )  VL(IV,L) = 0.
0137 *
0138     20 CONTINUE
0139     30 CONTINUE
0140 C
0141         GO TO 49
0142    42   ISTOP = 2
0143         GO TO 48
0144    47   CONTINUE
0145         ISTOP = 10
0146    48   CONTINUE
0147         LIST = L - 1
0148    49   CONTINUE
0149 *
0150 *
0151 *
0152 *
0153 *
0154        RETURN
0155        END
```

```
0001  OPT L,REL
0002  OPT LLE=80
0003  NAM INTERR
0004  IDNT INTERR
0005  XDEF INTERR
0006 ********** THIS PROGRAM ALLOWS REAL TIME INTERRUPS TO OCCUR **********
0007 *
0008  CSCT
0009 IFLAG RMB 2
0010 BYTE1 RMB 1
0011 BYTE2 RMB 1
0012 BYTE3 RMB 1
0013 BYTE4 RMB 1
0014 BYTE5 RMB 1
0015 BYTE6 RMB 1
0016 BYTE7 RMB 1
0017 BYTE8 RMB 1
0018 BYTE9 RMB 1
0019 BYTE10 RMB 1
0020 BYTE11 RMB 1
0021 BYTE12 RMB 1
0022 BYTE13 RMB 1
0023 BYTE14 RMB 1
0024 BYTE15 RMB 1
0025 BYTE16 RMB 1
0026 BYTE17 RMB 1
0027 BYTE18 RMB 1
0028 INTMSK RMB 2
0029 IRAM1 RMB 2
0030 IRAM2 RMB 2
0031 IRAM3 RMB 2
0032 IRAM4 RMB 2
0033 IRAM5 RMB 2
0034 IRAM6 RMB 2
0035 IRAM7 RMB 2
0036 IRAM8 RMB 2
0037 IRAM9 RMB 2
0038 IRAM10 RMB 2
0039 ITOP RMB 2
0040 IBOT RMB 2
0041 IDATA RMB 1500
0042 ************************
0043 *
0044 *  PROGRAM FOLLOWS
0045 *
0046  PSCT
0047 INTERR LDD #$E002
0048  EXG X,D       ;TO SAVE THE FORTRAN X REG AND USE IT AT E002 IN THIS PROG
0049  STD BYTE17    ;FORTRAN X REG IS SAVED
0050  ORCC #$10     ;SET IRG FLAG
0051  LDA #$04      ;THIS SET BIT 2 OF THE PIA CONTROL REG
0052  STA 0,X       ;PIA CONTROL REGISTER
0053  STA 4,X       ;PIA CONTROL REGISTER
0054  STA 5,X       ;PIA CONTROL REGISTER
0055  STA 8,X       ;PIA CONTROL REGISTER
0056  STA 9,X       ;PIA CONTROL REGISTER
0057  STA 12,Y      ;PIA CONTROL REGISTER
0058  STA 13,X      ;PIA CONTROL REGISTER
```

```
0059  STA 16,X      ;PIA CONTROL REGISTER
0060  STA 17,X      ;PIA CONTROL REGISTER
0061  LDA 1,X
0062  ORA #$07      ;TO CONFIGURE FOR INTERRUPTS THROUGH PIA
0063  STA 1,X
0064  LDD #$0000    ;SO IFLAG WILL BE RESET
0065  STD IFLAG
0066  LDD #$5620    ;ADDRESS OF INTERRUPT PROGRAM
0067  STD >$DFFE    ;PSEUDO IRQ INTERRUPT VECTOR
0068  LDD BYTE17    ;X REG STORED AT BYTE17
0069  TFR D,X       ;RESTORES THE X REG FOR FORTRAN
0070  LDD #$812E
0071  STD BYTE17    ;SET UP POINTER FOR IDATA STACK
0072  LDA >$E007
0073  ORA #$3B
0074  STA >$E007    ;TURNS ON THE SHIFT REGISTERS
0075  ANDCC #$EF    ;CLEARS INTERRUPT FLAG
0076  RTS
0077  END
```

```
PAGE 001  IOPKG   .SA:1            UMET-1                    15 DEC 1982

0001  NAM IOPKG
0002  XDEF IN$NP,OUTCH$,IN$NE,IN$NPE
0003  XDEF PCRLF$,PDAT1$,LOUTC$,INITLZ
0004  IDNT MDOS 6809 FORT LIB -  3.11 : MONITOR INDEP I/O PKG
0005  TTL MONITOR INDEP I/O PACKAGE
0006  SPC 1
0007 *****************************************************
0008 * Instructions to users:
0009 * ---------------------
0010 * This module allows use of monitor independent
0011 * subroutines with FORTRAN.  This is done as
0012 * follows:
0013 *  1. Change the I/O address definitions below
0014 *     to the correct values for your system.
0015 *  2. Assemble this file as follows:
0016 *       =RASM IOPKG;-FRXLN=80
0017 *  3. In "RLOAD", load this module (IOPKG.RO)
0018 *     BEFORE doing a FORTRAN library (FORLB.RO)
0019 *     search.
0020 *-----------------------------------------------------
0021 * If your monitor already has the following EXBUG
0022 * equivalent subroutines, then memory size may be
0023 * conserved by putting the following routine into
0024 * a seperate file (IOADRS) and changing the entry
0025 * locations.  Assemble this file as follows:
0026 *        =RASM IOADRS;RXLN=80
0027 *
0028 *  NAM IOADRS
0029 *  TTL I/O ADDRESSES
0030 *  IDNT MDOS 6809 FORT- RESIDENT MONITOR I/O EQUATES
0031 *  XDEF IN$NP,OUTCH$,PCRLF$,PDAT1$,LOUTC$
0032 *  SPC 1
0033 * IN$NP  EQU $F015 INPUT 1 CHAR, NO PARITY, WITH ECHO
0034 * OUTCH$ EQU $F018 OUTPUT CHAR (SPEED FILL)
0035 * PCRLF$ EQU $F021 PRINT CR, LF (USES A)
0036 * PDAT1$ EQU $F027 PRINT DATA STRING (X --> STRING)
0037 * LOUTC$ EQU $EBCC PRINT CHAR ON LPR (A= CHAR)
0038 * *                        IF ERROR, CARRY SET ON RETURN
0039 *  END
0040 *
0041 * Load this module (IOADRS.RO) before doing a FORTRAN
0042 * library (FORLB.RO) search.  Or you may specify the
0043 * addresses at link time via the "DEF" command before
0044 * doing a FORTRAN library search.
0045 *   Example:
0046 *        =RLOAD
0047 *        ?DEF:IN$NP=$F015
0048 *        ?DEF:OUTCH$=$F018
0049 *        ?DEF:PCRLF$=$F021
0050 *        ?DEF:PDAT1$=$F027
0051 *        ?DEF:LOUTC$=$EBCC
0052 *        ?BASE
0053 *           etc.
0054 *-----------------------------------------------------
0055  PAGE
0056 ***** EQUATES ******
0057 CR     EQU $0D
0058 LF     EQU $0A
```

```
0059 EOT    EQU $04
0060 SKIP2  EQU $8C SKIP 2 LOCATIONS
0061 *                          (CPX # OF CODE)
0062  SPC 1
0063 ***** I/O ADDR DEFS ******
0064 ACIAR$ EQU $EC14 Resident monitor's ACIA adr
0065 .ACIAI EQU ACIAR$ Input ACIA base address
0066 *                          $FCF4= EXBUG compatible
0067 *                          +0= Status,  +1= Data
0068 .CTRLI EQU %00010001 Input ACIA ctrl reg byte
0069 *     * Divide by 16 clock
0070 *     * 8-bits + 2 stop bits
0071 *     * RTS low, interrupts disabled
0072  SPC 1
0073 .ACIAO EQU ACIAR$ Output ACIA base address
0074 *                          $FCF4= EXBUG compatible
0075 *                          +0= Status,  +1= Data
0076 .CTRLO EQU %00010001 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 %00111100 LP "A" ctrl reg byte
0088 .CTRLB EQU %00111100 LP "B" ctrl reg byte
0089 .STRBA EQU %00110100 LP "A" ctrl reg strobe
0090 *            $3C,$3C,$34= EXBUG/Centronics compatible
0091 * Note: If a serial line printer is to be used, make
0092 *   the following changes:
0093 *     1) Change .LPIA to .LACIA and substitute
0094 *        proper address.
0095 *     2) Change .CTRLA to .ACIAL and substitute
0096 *        proper control reg byte.
0097 *     3) Delete .CTRLB and .STRBA lines.
0098 *     4) Replace the LPOUTC subroutine with a
0099 *        matching serial driver.
0100 *     5) Change INITLZ subroutine starting at
0101 *        INIT3 to initialize the line printer
0102 *        ACIA.
0103  PAGE
0104 * Monitor I/O address table for FORTRAN I/O
0105 * named common PSCT so user can easily overlay
0106 * with other I/O addresses by loading last.
0107  SPC 1
0108 .IOADR COMM PSCT
0109 ACIAI$ FDB .ACIAI Input ACIA address
0110 CTRLI$ FCB .CTRLI Input ACIA ctrl reg byte
0111 ACIAO$ FDB .ACIAO Output ACIA address
0112 CTRLO$ FCB .CTRLO Output ACIA ctrl reg byte
0113 LPIA$  FDB .LPIA  Lineprinter PIA address
0114 CTRLA$ FCB .CTRLA LP PIA ctrl reg A byte
0115 CTRLB$ FCB .CTRLB LP PIA ctrl reg B byte
0116 STRBA$ FCB .STRBA LP PIA ctrl reg A strobe
```

```
0117  SPC 1
0118 .CNNUL COMM PSCT
0119 CHNUL RMB 1 NUMBER OF NULLS AFTER EACH CHAR.
0120 CRNUL RMB 1 NUMBER OF NULLS AFTER EACH CR/LF.
0121  SPC 1
0122 .CIC COMM DSCT
0123 CIC$ RMB 1
0124  SPC 1
0125 .INIT COMM PSCT
0126  FDB INITLZ OVERLAY ADR IN ET$R0A
0127  SPC 1
0128 .M12CA COMM PSCT
0129 GTMAIN FDB 0 OVERLAID BY MM12 IF PRESENT
0130  PAGE
0131  PSCT
0132 ***********************************************
0133 * Initialize FORTRAN I/O Subroutine
0134 *    * Called by FORTRAN main program.
0135 *    * Does not assume reset occurred.
0136 ***********************************************
0137 INITLZ LDX ACIAO$ Get output ACIA address
0138  CPX #ACIAR$ Resident ACIA?
0139  BEQ INIT2 Yes, skip output init.
0140  LDX #0
0141 INIT1 BSR RTRN Delay before master reset
0142  DEX .  (may not be required)
0143  BNE INIT1
0144  LDX ACIAO$
0145  LDAB CTRLO$
0146  BSR .INITA Initialize output ACIA
0147 INIT2 LDX ACIAI$ Get input ACIA address
0148  CPX #ACIAR$ Resident ACIA?
0149  BEQ INIT3 Yes, skip input init.
0150  LDAB CTRLI$
0151  BSR .INITA Initialize input ACIA
0152 INIT3 LDX LPIA$ Init. LP PIA
0153  CLRA
0154  STAA 1,X Clear control registers
0155  STAA 3,X . in case no reset
0156 *                          (accesses data dir reg)
0157  STAA 2,X Set B DDR for inputs (=$00)
0158  COMA
0159  STAA 0,X Set A DDR for outputs (=$FF)
0160  LDAA CTRLA$
0161  STAA 1,X Set A control reg
0162  LDAA CTRLB$
0163  STAA 3,X Set B control reg
0164 RTRN RTS
0165  PAGE
0166 ***********************************************
0167 * Initialize ACIA Subroutine
0168 *    Entry:  X= base ACIA address
0169 *            B= control register byte
0170 ***********************************************
0171 .INITA LDAA #3
0172  STAA 0,X Master reset
0173  STAB 0,X Set control register
0174  RTS
```

```
0175  SPC 1
0176 ********************************************
0177 *  Print CR,LF,NULL Subroutine
0178 *     uses A register
0179 *     preserves B and X
0180 ********************************************
0181 PCRLF$ PSHB
0182  LDAA #CR
0183  BSR OUTCH$
0184  LDAA #LF
0185  BSR OUTCH$
0186  PULS B,PC RETURN
0187  SPC 1
0188 ********************************************
0189 * Input char, strip parity, & echo subroutine
0190 *     Exit:  A= char (7-bits)
0191 *            B and X preserved
0192 ********************************************
0193 IN$NP JSR IN$NPE
0194 * Fall into char output subroutine
0195  SPC 1
0196 ********************************************
0197 * Character Output Subroutine
0198 *    A= Char to output (8-bits)
0199 *    A, B and X preserved
0200 ********************************************
0201 OUTCH$ PSHS A,B,X Save reg's
0202 OUTCH1 LDX ACIA0$
0203  LDAB 0,X Get ACIA status
0204  ANDB #2 Ready yet?
0205  BEQ OUTCH4 No, try again
0206  LDA 0,S
0207  STAA 1,X Yes, send char
0208  LDAB CHNUL Char nulls
0209 * Test for CR nulls or char nulls
0210  CMPA #CR
0211  BNE OUTCH5
0212  LDAB CRNUL CR nulls
0213 OUTCH5 TSTB
0214  BEQ OUTCH7
0215 OUTCH6 LDAA 0,X Get ACIA status
0216  ANDA #2 Ready yet?
0217  BEQ OUTCH6 No, try again
0218  CLRA
0219  STAA 1,X Send null
0220  DECB
0221  BRA OUTCH5
0222  SPC 1
0223 OUTCH7 PULS A,B,X,PC Restore reg's & return
0224  SPC 1
0225 OUTCH4 LDX GTMAIN
0226  BEQ OUTCH1
0227  TST CIC$ MM12 Controller in charge?
0228  BNE OUTCH1 Yes (or no MM12 present)
0229  JSR 0,X Do possible handshake
0230  BRA OUTCH1
0231  PAGE
0232 ********************************************
```

```
0233 * Print Message String Subr.(no CR,LF)
0234 *     X= Adr of data string
0235 *     uses A and X reg's
0236 *     preserves B
0237 ***********************************************
0238 PDAT1$ LDAA 0,X Get next char
0239  CMPA #EOT
0240  BEQ RTRN
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)
0248 *            B and X preserved
0249 ***********************************************
0250 IN$NE PSHS X
0251 INCH0 LDX ACIAI$
0252  LDAA 0,X Get ACIA status
0253  ASRA Ready?
0254  BCC INCH2
0255  LDAA 1,X Yes, get char
0256  PULS X,PC Return
0257  SPC 1
0258 INCH2 LDX GTMAIN
0259  BEQ INCH0
0260  TST CIC$
0261  BNE INCH0
0262  PSHB
0263  JSR 0,X
0264  PULB
0265  BRA INCH0
0266  PAGE
0267 ***********************************************
0268 * Output Char to LP Subroutine
0269 *    Entry: A= char to print  (8-bits)
0270 *    Exit:  C= 1 if error
0271 *           A, B and X preserved
0272 ***********************************************
0273 LOUTC$ PSHS X
0274  LDX GTMAIN
0275  BEQ LOUT0
0276  TST CIC$
0277  BEQ LOUT10
0278 LOUT0 LDX LPIA$
0279  STAA 0,X Send data
0280  LDAA 0,X Clear acknowledge
0281  LDAA STRBA$
0282  BSR STROBE Send strobe
0283 LOUT1 LDAA 2,X Check status
0284  ANDA #3 Bit 0=Select,
0285 *                          Bit 1=Paper empty
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
```

```
0291 LOUTX LDAA 0,X
0292  PULS X,PC Return
0293  SPC 1
0294 LOUTER SEC
0295  BRA LOUTX
0296  SPC 1
0297 LOUT10 PSHS A,B
0298  JSR 0,X
0299  PULS 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 *            B and X preserved
0316 ***********************************************
0317 IN$NPE BSR IN$NE
0318  ANDA #$7F Strip parity
0319  RTS
0320  PAGE
0321  END
```

```
0001  OPT L,REL
0002  OPT LLE=80
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 ITOP RMB 2
0039 IBOT 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 >$E003       ;TURNS OFF INTERRUPTS
0050  JMP >$F830      ;JUMPS TO MONITOR
0051  END
```

```
PAGE 001  MAIN     .SA:1              UMET-1                        15 DEC 1982

0001 *                DECOMMUTATOR PROGRAM - UNIV. OF UTAH
0002 C
0003        OPTION STACK=200
0004 C
0005 *
0006 ****************************** COMMON BLOCK ******************************
0007 *
0008        INCLUDE 'COMMON.SA'
0009 C
0010 ***************************************************************************
0011 *
0012       DIMENSION LINE(40),LDATE(3),LTIME(3),ID(3),ISTT1(3),ISTT2(3)
0013 *
0014 *
0015 *            SEARCH AND GATE HALFWIDTHS
0016 *
0017 C
0018 *
0019        WRITE(100,5431)
0020  5431 FORMAT(////////////////,31(' '),'UNIVERSITY OF UTAH',/,
0021 &       35('*'),'  UMET-1  ',34('*'),//,'          ',
0022 &       'REAL TIME PROCESSING OF METEOROLOGICAL BALLOON SOUNDING DATA',
0023 &       /////////)
0024        WRITE(101,5432)
0025  5432 FORMAT('ENTER PRINTER OUTPUT CODE',/,'  1=NO DIAGNOSTICS',/,
0026 &       '  2=CONDENSED POINTS AFTER FLIGHT',/,
0027 &       '  3=DIAGNOSTICS PRINTED DURING FLIGHT',/)
0028        READ(101,5433)IDG
0029 5433  FORMAT(I3)
0030 *
0031 *
0032        DO 13 J=1,120
0033        DO 14 I=1,7
0034         VL(I,J) = 0.0
0035  14     CONTINUE
0036  13     CONTINUE
0037 C
0038 C
0039        CALL PRNTER      ;SET UP PRINTER FOR 132 CHAR LINE
0040        IOUT=102
0041        WRITE(IOUT,9998)
0042  9998 FORMAT('1',131('*'),/,10('*'),41X,11('.'),' UMET-1 ',11('.'),
0043 &         40X,10('*'),/,10('*'),38X,'   CONDENSER',
0044 &        '/DECOMMUTATOR PROGRAM   ',37X,10('*'),/,10('*'),
0045 &        111X,10('*'),/,131('*'))
0046         WRITE(IOUT,9997)
0047  9997 FORMAT(//103X,'UNIV. OF UTAH  AUG 1982',//)
0048        WRITE(IOUT,999)
0049  999 FORMAT (1X,45X,'********** INPUT  DATA **********'//)
0050 C      RUNNING TIME, T, IS SECONDS ELAPSED AFTER LAUNCH.
0051 C       *** TPROC IS THE TIME INTERVAL (SEC)TO BE PROCESSED,
0052 C       *** BEGINNING AT TSTART SECONDS AFTER LAUNCH.
0053 C           TLANCH = BALLOON RELEASE TIME OF DAY. COUNTED IN SECONDS
0054 *
0055  5008 TPROC=9999.0
0056        TSTART=0.
0057        GOTO 5108
0058 1003  WRITE(IOUT,10004)I1,I2,TS3
```

```
0059 10004  FORMAT(' BALLOON RELEASED AT ',2(I2,':'),F4.1)
0060 C
0061 C      CONVERT I1(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 *
0069 C       *** LINENO IS THE NUMBER OF LINES PER PAGE TO BE PRINTED
0070 *
0071 5108    LINENO=54
0072 *
0073 *
0074 C           INITIALIZE CONDENSER
0075 *
0076        FSUM = 0.0
0077        NSUM = 0
0078 *
0079 C           LOSS OF SIGNAL FLAG LOS
0080 *
0081        LOS = 0
0082 *
0083 C           MODE INTERVAL ( OVERLAPPING BANDS) HALF-WIDTH ( 0.5 HZ)
0084 *
0085        HGATE = 1.0
0086 *
0087 C           SIGNAL RANGE ( SIGMIN TO SIGMAX HZ )
0088 *
0089        SIGMIN = 5.
0090        SIGMAX = 205.
0091        IN = (( SIGMAX - SIGMIN )/ HGATE ) + 1
0092 *
0093 C           CONDENSED DATA INDEX JK, FOR ONE DECOMMUTATION CYCLE
0094 *
0095        JK=0
0096        ISTPER=0
0097 C
0098        DO 1 I=1,900
0099        COND(1,I) = 0.0
0100        COND(2,I) = 0.0
0101        COND(3,I) = 0.0
0102        ICOND(1,I) = 0
0103        ICOND(2,I) = 0
0104  1     CONTINUE
0105 C
0106 *
0107 C           INITIAL EXPECTED SIGNAL LEVELS
0108 *
0109 *
0110 *
0111        JJ=0
0112 *
0113 *                 INITIALIZE TABLE
0114 *
0115         TNOH = 10000.0
0116         VL(1,1) = 0.0
```

```
0117          DLIST=60.
0118          TGMDAQ=20.
0119          LIST=1
0120         WRITE(100,9811)
0121 9811    FORMAT('ENTER STATION ID (XX)',/)
0122         READ(101,5433)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(101,5433)IAZU
0129         WRITE(100,9813)
0130 9813    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(100,9815)
0136 9815    FORMAT('ENTER LAUNCH YEAR (XX)',/)
0137         READ(101,5433)IYEAR
0138         WRITE(100,9802)
0139 9802    FORMAT('ENTER SURFACE TEMPERATURE IN C  (XX.X)',/)
0140         READ(101,5445)SURT
0141         WRITE(100,9803)
0142 9803    FORMAT('ENTER SURFACE RELATIVE HUMIDITY % (XX.X)',/)
0143         READ(101,5445)SURH
0144         WRITE(100,9801)
0145 9801    FORMAT('ENTER SURFACE PRESSURE IN MBAR (XXXX.X)',/)
0146         READ(101,5444)PP0
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 5459    FORMAT(I5)
0157         WRITE(100,9804)
0158 9804    FORMAT('ENTER UNADJUSTED REFERENCE ORDINATES (XX.X)',/)
0159         READ(101,5445)PR0
0160         WRITE(100,9817)
0161 9817    FORMAT('ENTER AIR TEMP CALIBRATION ORDINATES (XXX.X)',/)
0162         READ(101,5467)RECTP
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)FTEMP0
0169         WRITE(100,6137)
0170 6137    FORMAT('ENTER SURFACE RH ORDINATES (XXX.X)',/)
0171         READ(101,5467)FRH0
0172 5444    FORMAT(F6.1)
0173 5445    FORMAT(F4.1)
0174 5467    FORMAT(F5.1)
```

```
0175        WRITE(100,9819)
0176 9819   FORMAT('FOR MILITARY SONDE (ML419,ML418) ENTER 1',/,
0177 &             'FOR STANDARD NWS SONDE ENTER 2',/)
0178        READ(101,5433)ICRBN
0179        IF (FR0 .LT. .01) GOTO 6665
0180        RFL=2.0*FR0-10.0
0181        PFL=RFL+20
0182        GOTO 6666
0183 6665   FR0=95.0
0184        RFL=170.
0185 6667   PFL=190.
0186 6666   V2( 4) = FP0
0187        V2( 5) = FR0*2.
0188        V2( 6) = FTEMP0
0189        V2( 7) = FRH0
0190 C
0191 C                SET INITIAL FREQUENCY GATES FROM SURFACE ORDINATE INPUTS
0192 C
0193         CNVOF = 2.*FR0/95.
0194         IF (CNVOF .LT. .01) GO TO 5
0195         TF = FTEMP0*CNVOF
0196         HF=FRH0*CNVOF
0197       5 CONTINUE
0198 *
0199 *                MANUAL BURST INPUTS
0200 *
0201        TBRST = 1.E22
0202
0203        WRITE(100,9805)
0204 9805   FORMAT('ENTER CONTACT NUMBER AT BURST (XXX.X)',/,
0205 &             'IF < 10 IS INPUT, VALUE DEFAULTS TO 180.0',/)
0206        READ(101,5446)CBRST
0207 5446   FORMAT(F5.1)
0208        IF(CBRST .LT. 10.)CBRST = 180.01
0209        HEIGHT=HEIGHT+.0009     ;TO ACCOUNT FOR TURNCATION
0210        FP0=FP0+.0009
0211        FTEMP0=FTEMP0+.0009
0212        FR0=FR0+.0009
0213        RECTF=RECTF+.0009
0214        CALRH=CALRH+.0009
0215        WRITE(IOUT,1028)IDSTAT,HEIGHT,IAZU,IMONTH,IDAY,IYEAR,SURT,SURH,
0216 &            FP0,VSFC,DSFC,IDSOND,FR0,RECTF,CALRH,FTEMP0,FRH0,ICRBN,
0217 &            CBRST
0218        HEIGHT=HEIGHT-.0009      ;RESTORE INPUT VALUES
0219        FP0=FP0-.0009
0220        FTEMP0=FTEMP0-.0009
0221        FR0=FR0-.0009
0222        RECTF=RECTF-.0009
0223        CALRH=CALRH-.0009
0224  1028  FORMAT('STATION ID =',I3,/,'STATION HEIGHT =',F6.1,/,
0225 &             'ZERO AZIMUTH =',I2,/,
0226 &             'DATE =',I2,'-',I2,'-',I2,/,'SURFACE TEMP =',F6.1,/,
0227 &             'SURFACE RH =',F6.1,/,
0228 &             'SURFACE PRESS =',F6.1,/,'SURFACE WIND SPEED =',F6.1,/,
0229 &             'SURFACE WIND DIRECTION =',F6.1,/,'SONDE ID =',I5,/,
0230 &             'REFERENCE ORD =',F6.1,/,'TEMP CAL ORD =',F6.1,/,
0231 &             'RH CAL ORD =',F6.1,/,'SURFACE TEMP ORD =',F6.1,/,
0232 &             'SURFACE RH ORD =',F6.1,/,'SONDE TYPE =',I4,/,
```

```
0233 &              'BURST CONTACT = ',F6.1)
0234 *
0235 *
0236          ISTOP = 0
0237 *
0238 C                PRESSURE CALIBRATION INPUT
0239 *
0240 1623    WRITE(100,8201)
0241 8201    FORMAT('ENTER PCAL INPUT CODE',/,'  1=USING TAPE READER',/,
0242 &               '  2=ENTERING BY HAND',/)
0243         READ(101,5433)IDUMM
0244         IF(IDUMM .EQ. 2)GOTO 1625
0245         IFLAG=0
0246         DO 12 J=1,180
0247         PCAL(J)=0.0
0248  12     CONTINUE
0249         CALL SETUP
0250         WRITE(100,1626)
0251 1626    FORMAT('ENTER THE PUNCH PAPER TAPE')
0252         ISTPER=0
0253         J=0
0254 1627    IF(IFLAG .EQ. 1)GOTO 1628
0255         CALL STOPER
0256         IF(ISTPER .EQ.1)GOTO 1625
0257         GOTO 1627
0258 1628    J=J+1
0259         PCAL(J)=IRAM(1)+(.1*IRAM(2))
0260         IFLAG=0
0261         IF(PCAL(J) .LT. .1)GOTO 9881
0262         IF(J .GE. 179)GOTO 9881
0263         GOTO 1627
0264 9881    CALL OFF
0265         GOTO 9882
0266 1625    CONTINUE
0267         DO 988 I=1,180
0268         WRITE(100,9809)I
0269 9809    FORMAT('ENTER PCAL',I3)
0270         READ(101,5444)PCAL(I)
0271  988    CONTINUE
0272  9882   IF(PCAL(1) .LT. .01)GOTO 1623
0273         DIFF1=PCAL(1)-PCAL(2)
0274         DO 8 I=2,179
0275          IST=I/20
0276          PERC=.11+IST/100.
0277          DIFF2=PCAL(I)-PCAL(I+1)
0278          IF(PCAL(I) .GT. 0.0)GOTO 2
0279          IF(PCAL(I+1) .LT. 0.01)GOTO 9
0280  2       DIFFAV=(DIFF1+DIFF2)/2.0
0281          DIFFHI=DIFFAV*(1. + PERC)
0282          DIFFLO=DIFFAV*(1. - PERC)
0283          IF(DIFF2 .GT. DIFFHI .OR. DIFF2 .LT. DIFFLO)GOTO 6
0284  9       DIFF1=DIFF2
0285          GOTO 8
0286  6       IF(PCAL(I)-DIFF1 .LT. 0)GOTO 8
0287          DUMMY=PCAL(I)-DIFF1
0288          IDUMMY = I + 1
0289          WRITE(IOUT,100)IDUMMY,PCAL(I+1),DUMMY
0290  100     FORMAT(7X,' PCAL(',I3,') WAS',F10.1,' AND IS NOW',F10.1)
```

```
PAGE 006  MAIN    .SA:1            UMET-1                    15 DEC 1982

0291          PCAL(I+1)=PCAL(I)-DIFF1
0292  8       CONTINUE
0293          WRITE(IOUT,10024)
0294 10024    FORMAT('1','BAROSWITCH PRESSURE CALIBRATION TABLE')
0295 10020    FORMAT(8F10.1)
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(1X,I3,':  ',8F10.1)
0305  30      CONTINUE
0306          WRITE(IOUT,10026) (PCAL(IX),IX=177,180)
0307 10026   FORMAT(1X,'177:  ',4F10.1)
0308         DO 1675 I=1,180
0309         PCAL(I)=PCAL(I)-.05       ;RESTORES VALUES OF PCAL TABLE
0310 1675    CONTINUE
0311         DO 15 JP = 1,180
0312  15      IF(PCAL(JP) .LT. FP0)GO TO 16
0313  16      ICR0 = ((FP0 - PCAL(JP-1))/(PCAL(JP)-PCAL(JP-1)))*100.
0314         ICR0 = ICR0 + (JP-1)*100
0315
0316          DO 3 LCNTK = 1,180
0317     3    IF(PCAL(LCNTK) .LT. 0.1) GO TO 4
0318     4    LCNTK = LCNTK - 1
0319         DUMMY = ICR0            ;CHANGES TO REAL
0320         AICR0 = DUMMY/100.
0321         WRITE(IOUT,10016)AICR0,LCNTK
0322 10016 FORMAT(' EFFECTIVE CONTACT NUMBER AT LAUNCH = ',F6.2,/
0323 &        ' HIGHEST CONTACT NUMBER CALIBRATED = ',I3)
0324 *
0325           WRITE(100,6789)
0326 6789      FORMAT('IF PCAL TABLE IS OK ENTER 1',/,'IF NOT OK ENTER 2',/)
0327           READ(101,5433)IDUM
0328           IF(IDUM .EQ. 2)GOTO 1623
0329 *
0330 6786      WRITE(100,6787)
0331 6787      FORMAT('ENTER HOUR OF LAUNCH (XX)',/)
0332           READ(101,5433)I1
0333           WRITE(100,6788)
0334 6788      FORMAT('ENTER MIN OF LAUNCH (XX)',/)
0335           READ(101,5433)I2
0336           WRITE(100,6790)
0337 6790      FORMAT('ENTER SECOND OF LAUNCH (XX.X)',/)
0338           READ(101,5445)TS3
0339           GOTO 1003
0340 *
0341 6764      WRITE(100,6791)
0342 6791      FORMAT('IF LAUNCH TIME IS OK ENTER 1',/,'IF NOT OK ENTER 2',/)
0343           READ(101,5433)IDUM
0344           IF(IDUM .EQ. 2)GOTO 6786
0345 C
0346 C             FIND TSTART IN RAW DATA
0347         IF(IDG .EQ. 1)IOUT=101
0348         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 *
0359 40      CONTINUE
0360 *
0361 ***************  HANDLE REAL TIME DATA ********************************
0362 *
0363 *
0364        DO 41 J=6,10
0365 4343   IF(IFLAG .GT. 0)GOTO 4344
0366        GOTO 4343                ;LOOK AT FLAG UNTIL SET (IE DATA RECIEVED)
0367 4344   INTMSK=1                 ;MASK INTERRUPTS
0368        IFLAG=IFLAG-1
0369        IF(IBOT .EQ. 150)IBOT=0
0370        IDUM=IBOT*10             ;THERE ARE 10 IDATAS FOR EACH DATA POINT
0371        IBOT=IBOT +1             ;WORK WAY UP STACK OF DATA
0372        DUMMY=IDATA(IDUM+9)      ;PART OF MET WORD
0373        DUMMY1=IDATA(IDUM+10)    ;REST OF MET WORD
0374        FREQ(J)=(DUMMY*1000)+DUMMY1
0375        DUMMY=IDATA(IDUM+6)      ;PART OF EL WORD
0376        DUMMY1=IDATA(IDUM+5)
0377        EL(J)=DUMMY1+(DUMMY*.01)
0378        DUMMY=IDATA(IDUM+8)
0379        DUMMY1=IDATA(IDUM+7)
0380        AZ(J)=DUMMY1+(DUMMY*.01)
0381        DUMMY=IDATA(IDUM+4)
0382        DUMMY1=IDATA(IDUM+3)
0383        SECS=DUMMY1+(DUMMY*.1)
0384        DUMMY=IDATA(IDUM+2)
0385        AMINU=(DUMMY*60.)+SECS
0386        HOURS=(IDATA(IDUM+1))*3600
0387        TIME(J)=HOURS+AMINU-TLANCH ;TIME IS IN SECONDS FROM LAUNCH
0388        INTMSK=0
0389 41     CONTINUE
0390        WRITE(100,2003)TIME(10)
0391 2003   FORMAT(F7.2,' SECONDS FROM LAUNCH')
0392        IF(TIME(10) .LT. 0.)GOTO 40
0393        WRITE(100,9825)
0394 9825   FORMAT('**** PAST LAUNCH TIME *****')
0395 *
0396 ******************** REAL TIME DATA IN PROGRAM ***************************
0397        JJ = JJ+ 5
0398 C
0399 C
0400 99      CALL ADVANC (JJ,TSTOP,TLANCH,TGMDAQ,MASK)
0401         IF (MASK .EQ. 1) GOTO 81
0402         IF (MASK .EQ. 2) GOTO 82
0403         IF (MASK .EQ. 3) GOTO 825
0404         IF (MASK .EQ. 4) GOTO 826
0405 C
0406         JKMEM = JK
```

```
0407         CALL TRACK(LOS,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  410     CALL DECOM(JK,TNOH,TF,HF,ICRO,RFL,PFL,IDG)
0413 C
0414         IF ( COND(1,JK) .GT. TERST) GO TO 84
0415         GO TO 99
0416 2204   FORMAT (' ERROR IN CONDPASS#. READ')
0417 C*******************************************************************
0418 C
0419 C
0420   825   WRITE(IOUT,1804)
0421         ISTOP=11
0422         GOTO 90
0423   826   WRITE(IOUT,1806)
0424         ISTOP=12
0425         GOTO 90
0426   81    WRITE(IOUT,1810) TSTOP,TIME(10),JJ
0427         ISTOP = 7
0428        GO TO 90
0429   82    WRITE(IOUT,1820) TIME(10), JJ
0430         ISTOP = 6
0431         GO TO 90
0432    83 WRITE(IOUT,1830) LOS
0433         ISTOP = 5
0434        GO TO 90
0435  85    WRITE(IOUT,1850)
0436        ISTOP = 8
0437        GO TO 90
0438  84     WRITE(IOUT,1840)JK,COND(1,JK),TERST
0439  1810 FORMAT(2X, '       TSTOP,TIME(10),JJ =',     2F10.1, 1I10)
0440  1820 FORMAT(2X, 'END OF FILE,TIME(10),JJ =', 10X, F10.1, 1I10)
0441  1830 FORMAT (2X, ' LOS = ',10X,I6)
0442  1840  FORMAT(/,' TIME EXCEEDS TERST....COND(1,',I3,') =',F10.2,
0443 &         ' > TERST =',F10.2)
0444  1850 FORMAT(' EXCEEDED COND ARRAY DIMENSION')
0445  1804 FORMAT('FLIGHT TIME GREATER THAN 110 MIN.')
0446  1806 FORMAT('STOPPED BY OPERATOR')
0447 *
0448 C
0449 C
0450 C      LIST UNINTERPOLATED COND/ICOND MATRIX
0451 *
0452  90     WRITE(IOUT,1900)
0453  1900   FORMAT(//,' CONDENSER DONE.'/' DECOMMUTATOR DONE.'/,
0454 &        ' INTERPOLATION FOLLOWS.......')
0455
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. 200)ICOND(2,JC) = 0
0461          INDAX = ICOND(1,JC)
0462         IF(ICOND(1,JC).GE.10.AND.ICOND(1,JC).LE.19)ICOND(1,JC) = 1
0463         IF(ICOND(1,JC).GE.40.AND.ICOND(1,JC).LE.49)ICOND(1,JC) = 4
0464          IF (ICOND(1,JC).EQ.0.OR.ICOND(1,JC).GT.5)ICOND(1,JC) = 5
```

```
0465
0466
0467  95   FORMAT('1',' JK',6X,'TIME: OF DAY;    ELAPSED',
0468 &      '  DWELL     TEMP.      REF. HIGH REF.  REL.HUM.   UNDECOM',
0469 &       ' BAROSWITCH   WORKING   WORKING',
0470 &      /6X,' HH:MM:SS.S    HOURS MM:SS.S  (SEC)',21('-'),
0471 &      '--(HZ)--',21('-'),'  CONTACT #   CONTACT   CHANNEL',/)
0472       RT2 = (COND(1,JC)+TLANCH)/3600.
0473       RT4 = COND(2,JC)
0474       ELPT = COND(1,JC)
0475       IT3 = ELPT/60
0476       DUMMY =ELPT/60.
0477       IDUMMY=DUMMY                    ;TRUNCATES VALUE
0478       DUMMY1=IDUMMY                   ;TRUNCATED REAL
0479       DUMMY=DUMMY-DUMMY1
0480       RT3=DUMMY*60.                   ;GIVES THE REMAINDER
0481       IT1 = RT2                       ;CHANGES TO INTEGER
0482       IT2 = (RT2-IT1)*60
0483       RT1 = ((RT2-IT1)*60-IT2)*60.
0484
0485       IDUMMY = JC-1
0486       IF(MOD(IDUMMY,LINENO) .EQ.0)WRITE(IOUT,95)
0487       IDUMMY=ICOND(1,JC)
0488        GO TO (1910, 1920, 1930, 1940, 1950) IDUMMY
0489  1910 WRITE(IOUT,191)JC,IT1,IT2,RT1,RT2,IT3,RT3,RT4,COND(3,JC),
0490 &       ICOND(2,JC),IBC,INDAX
0491       GO TO 196
0492  1920 WRITE(IOUT,192)JC,IT1,IT2,RT1,RT2,IT3,RT3,RT4,COND(3,JC),
0493 &       ICOND(2,JC),IBC,INDAX
0494       GO TO 196
0495  1930 WRITE(IOUT,193)JC,IT1,IT2,RT1,RT2,IT3,RT3,RT4,COND(3,JC),
0496 &       ICOND(2,JC),IBC,INDAX
0497       GO TO 196
0498  1940 WRITE(IOUT,194)JC,IT1,IT2,RT1,RT2,IT3,RT3,RT4,COND(3,JC),
0499 &       ICOND(2,JC),IBC,INDAX
0500       GO TO 196
0501  1950 WRITE(IOUT,195)JC,IT1,IT2,RT1,RT2,IT3,RT3,RT4,COND(3,JC),
0502 &       ICOND(2,JC),IBC,INDAX
0503   196 CONTINUE
0504   191  FORMAT(1X,I3,3X,I2,':',I2,':',F4.1,F9.4,1X,I2,':',F4.1,F7.1,
0505 &         F10.2,40X,5X,I6,2I10)
0506   192  FORMAT(1X,I3,3X,I2,':',I2,':',F4.1,F9.4,1X,I2,':',F4.1,F7.1,
0507 &      10X,F10.2,30X,5X,I6,2I10)
0508   193  FORMAT(1X,I3,3X,I2,':',I2,':',F4.1,F9.4,1X,I2,':',F4.1,F7.1,
0509 &      20X,F10.2,20X,5X,I6,2I10)
0510   194  FORMAT(1X,I3,3X,I2,':',I2,':',F4.1,F9.4,1X,I2,':',F4.1,F7.1,
0511 &      30X,F10.2,10X,5X,I6,2I10)
0512   195  FORMAT(1X,I3,3X,I2,':',I2,':',F4.1,F9.4,1X,I2,':',F4.1,F7.1,
0513 &      40X,F10.2,5X,I6,2I10)
0514
0515 C*****************************************************************
0516 *
0517       CALL INTERP(JK,TNOH,ISTOP,LCNTK)
0518 *
0519 C
0520 C*****************************************************************
0521 *
0522 *
```

```
0523 *               PRINT TABLE
0524 *
0525        WRITE(IOUT,1030)
0526  1030 FORMAT ('1',  13X,'DECOMMUTATED OUTPUT AT UNIFORM TIME INTERVALS')
0527 C
0528        WRITE(IOUT,1131)
0529  1131 FORMAT(5X,'INDEX     TIME AZIMUTH ELEVATION PRESSURE REF FREQ',
0530 &         5X,'TEMP  REL HUM')
0531        WRITE (IOUT,1230)
0532  1230 FORMAT(14X,'(SEC)    (DEG)    (DEG)     (MB)', 5X,
0533 &        '(HZ)----(ORDINATES)---')
0534 ***************************** WRITE THE TABLE ********************
0535         DO 6767  LL=1,LIST
0536         IDUMMY=LL-1
0537         WRITE(IOUT,1330)IDUMMY,VL(1,LL),VL(2,LL),VL(3,LL),VL(4,LL),
0538 &       VL(5,LL),VL(6,LL),VL(7,LL)
0539  6767 CONTINUE
0540 *
0541 *
0542  1330 FORMAT(1X,I9,7F9.1)
0543 ********************* FINISHED WRITING THE TABLE ******************
0544 C
0545 C      FINISH EXPLAINING AND END
0546 C
0547        WRITE(IOUT,1043)
0548  1043   FORMAT('*** REAL TIME PROCESSING COMPLETE ***  ')
0549 C
0550         IF(ISTOP .EQ. 2) WRITE(IOUT,1042) KNTCT,LCNTK
0551  1042  FORMAT(1X,'STOPPED AT CONTACT NR',I5,5X,'LAST NON ZERO FCAL NR',
0552 &             I5)
0553         IF(ISTOP .EQ. 5) WRITE(IOUT,1045)
0554  1045  FORMAT(1X,'STOPPED, CONDENSER UNABLE TO FIND SIGNAL. TOO NOISY')
0555         IF(ISTOP .EQ. 6) WRITE(IOUT,1046)
0556  1046  FORMAT(1X,'STOPPED, END OF INPUT DATA (EOF)')
0557         IF(ISTOP .EQ. 7) WRITE(IOUT,1047)
0558  1047  FORMAT(1X,'STOPPED, REACH TSTOP (TSTART+TPROC)')
0559         IF(ISTOP .EQ. 8) WRITE(IOUT,1048)
0560  1048  FORMAT(1X,'STOPPED, COND OVERFLOW')
0561         IF(ISTOP .EQ. 10) WRITE(IOUT,1050)
0562  1050  FORMAT(1X,'COMPLETED TO BURST')
0563         IF(ISTOP .EQ. 11) WRITE(IOUT,1051)
0564  1051  FORMAT(1X,'FLIGHT TIME IN EXCESS OF 120 MIN.')
0565         IF(ISTOP .EQ. 12) WRITE(IOUT,1052)
0566  1052  FORMAT(1X,'STOPPED BY OPERATOR')
0567 *
0568 *
0569 *               WRITE PRD COMPATIBLE TAPE
0570 *
0571 *
0572 *               DOUBLE CHECK FOR ZERO PRESSURE VALUES AT TOP OF FLIGHT
0573 *
0574        DO 501 LL=1,LIST
0575        J = LIST - LL + 1
0576        IF(VL(4,J) .GT. 0.0) GO TO 502
0577  501   CONTINUE
0578        WRITE(IOUT,6000)
0579        GO TO 503
0580  502   CONTINUE
```

```
0561        LIST = J
0562  503   CONTINUE
0563 *
0564        NCDS=LIST     ;PASS THIS NUNBER TO PRD
0565 *
0566 *              ONE MINUTE DATA IS PASSED IN VL ARRAY
0567 *
0568        IOUT=102      ;SO PRD OUTPUT GOES TO THE PRINTER
0569 *
0570        CALL JUMPER
0571 *
0572 *
0573  6000  FORMAT(1X,'UNABLE TO FIND NON ZERO PRESSURE - EXPECT ERROR ',
0574 &              'IN ECC-PRD PROGRAM')
0575 *
0576       STOP
0577       END
```

```
0001  OPT L,REL
0002  OPT LLE=80
0003  NAM OFF
0004  IDNT OFF
0005  XDEF OFF
0006 *****
0007 *****
0008  CSCT
0009 IDATA KMB 300
0010  PSCT
0011 OFF LDA #$06
0012  STA >$E017
0013  RTS
0014  END
```

```
0001  OPT L,REL
0002  OPT LLE=80
0003  NAM PRNTER
0004  IDNT PRNTER
0005  XDEF PRNTER
0006 *
0007 *       THIS PROGRAM SETS UP THE PRINTER TO OUTPUT 132 CHAR PER LINE
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 ITOP RMB 2
0039 IBOT 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 PRNTER JSR >$5B52      ;INIT PRINTER
0048  LDA #$1B
0049  JSR >$5C0A            ;SEND
0050  LDA #$14
0051  JSR >$5C0A            ;PRINTER SET UP FOR 132 CHAR PER LINE
0052  LDA >$E007
0053  ORA #$30
0054  ANDA #$F7
0055  STA >$E007             ;THIS DISABLES THE SHIFT REGISTER
0056  RTS
0057  END
```

```
0001  OPT L,REL
0002  OPT LLE=80
0003  NAM REAL
0004  IDNT REAL
0005  XDEF REAL
0006 *
0007 ********** THIS IS THE INTERRUP PROGRAM **********
0008 *       DATA FROM THE TRADT IS VALID ON THE PIA'S
0009 *       TO CAUSE AN INTERRUP. THIS PROGRAM READS THE DATA
0010 *       INTO BYTES WHERE IT IS FORMATTED INTO INTERGERS
0011 *       AND STORED IN IRAMS, WHERE FORTRAN SUB 'FORM' RE-FORMATS
0012 *       THE DATA INTO REAL NUMBERS AND STORES THEN IN TEMP
0013 *       STORAGE. THEN INCERMENTS IFLAG AND RETURNS TO THIS
0014 *       PROGRAM WHERE A RTI IS DONE.
0015 *         THE COND PROGRAM THEN IS LOOKING FOR IFLAG TO BE SET
0016 *       SO IT CAN TAKE THE DATA FROM THE TEMP STORAGE AND USE IT.
0017 *
0018  CSCT
0019 IFLAG RMB 2
0020 BYTE10 RMB 1
0021 BYTE9 RMB 1
0022 BYTE8 RMB 1
0023 BYTE7 RMB 1
0024 BYTE6 RMB 1
0025 BYTE5 RMB 1
0026 BYTE4 RMB 1
0027 BYTE3 RMB 1
0028 BYTE2 RMB 1
0029 BYTE1 RMB 1
0030 BYTE11 RMB 1
0031 BYTE12 RMB 1
0032 BYTE13 RMB 1
0033 BYTE14 RMB 1
0034 BYTE15 RMB 1
0035 BYTE16 RMB 1
0036 TEMPX RMB 2
0037 INTMSK RMB 2
0038 IRAM1 RMB 2
0039 IRAM2 RMB 2
0040 IRAM3 RMB 2
0041 IRAM4 RMB 2
0042 IRAM5 RMB 2
0043 IRAM6 RMB 2
0044 IRAM7 RMB 2
0045 IRAM8 RMB 2
0046 IRAM9 RMB 2
0047 IRAM10 RMB 2
0048 ITOP RMB 2
0049 IBOT RMB 2
0050 IDATA RMB 1500     ;IRAM ARE STACKED INTO THE IDATA ARRY TO BE PROCESSED
0051 **********************
0052 *
0053 *
0054 *  PROGRAM FOLLOWS
0055 *
0056  PSCT
0057 REAL LDU #$810C     ;COMMON STARTS AT 8100 SO THIS WILL PUT STACK INTO BYTES
0058  LDX #$E000         ;BOTTOM ADDRESS OF PIA'S
```

```
0059 *
0060 *           NOTE HERE THAT PIAS ARE HARD WIRED SO RS0 GOES TO ADDRESS A1
0061 *           AND RS1 GOES TO A0. THIS PUT THE DATA PORTS NEXT TO EACH
0062 *           IN MEMORY, TO USE OF THE D REGISTER.
0063 *
0064  LDD 0,X          ;START GETTING THE DATA
0065  PSHU D           ;STORES PART OF MET WORD AT BYTE1 & BYTE2
0066  LDD 4,X          ;GETS MORE OF THE MET WORD
0067  PSHU D           ;STORES MORE OF THE MET WORD AT BYTE3 &BYTE4
0068  LDD 8,X
0069  PSHU D
0070  LDD 12,X
0071  PSHU D
0072  LDD 16,X
0073  PSHU D           ;RAW DATA IS NOW STORED FROM BYTE1-BYTE10
0074 *
0075 *          FORMAT THE DATA INTO INTERGER 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 *
0088 *         DO NOT WRITE IN NEW DATA IF FORTRAN IS FORMATTING OLD DATA
0089 *
0090  LDD INTMSK
0091  CMPD #$01
0092  BNE DOIT
0093  LDD #$02
0094  STD INTMSK
0095  LBRA DONE
0096 *
0097 *          FIND VALUE OF HOURS
0098 DOIT LDB BYTE10
0099  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  BCC NOCAR
0120  LDA BYTE11
0121  LSLA
0122  ORA #$01
0123  STA BYTE12
0124  BRA LOWMIN
0125 NOCAR LDA BYTE11
0126  LSLA
0127  STA BYTE12
0128 LOWMIN LDA BYTE9
0129  ANDA #$78
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 *          START ON EL WORD
0160  LDA BYTE7
0161  ANDA #$0F
0162  STA BYTE11
0163  LDB BYTE7
0164  ANDB #$F0
0165  LSRB
0166  LSRB
0167  LSRB
0168  LSRB
0169  LDA #$0A
0170  MUL
0171  ADDB BYTE11
0172  STD IRAM5
0173 *           DO PART OF DEGREE EL
0174  LDA BYTE6
```

```
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  LDB BYTE5
0190  LSRB
0191  LSRB
0192  LSRB
0193  LSRB
0194  LDA #$64
0195  MUL
0196  STD BYTE13
0197  LDB BYTE5
0198  ANDB #$0F
0199  LDA #$0A
0200  MUL
0201  ADDD BYTE13
0202  STD BYTE13
0203  LDA #$00
0204  LDB BYTE4
0205  LSRB
0206  LSRB
0207  LSRB
0208  LSRB
0209  ADDD 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
```

```
0233  ANDB #$0F
0234  LDA #$0A
0235  MUL
0236  ADDB BYTE11
0237  STD IRAM9
0238  *           DO LAST OF MET WORD
0239  LDB BYTE2
0240  ANDB #$0F
0241  LDA #$64
0242  MUL
0243  STD BYTE13
0244  LDB BYTE1
0245  ANDB #$01
0246  STD BYTE11
0247  LDB BYTE1
0248  LSRB
0249  LSRB
0250  LSRB
0251  LSRB
0252  LDA #$0A
0253  MUL
0254  ADDB BYTE11
0255  ADDD BYTE13
0256  STD IRAM10
0257  *
0258  *             DATA IS FORMATTED AS INTEGERS
0259  *             NOW FORMAT INTO IDATA ARRY FOR USE IN MAIN
0260  *             MET PROGRAM.
0261  *
0262  LDD ITOP
0263  ADDD #$01
0264  CMPD #150
0265  BNE ISTOP
0266  LDD #$0000
0267 ISTOP STD ITOP
0268  CMPD IBOT
0269  BNE SETFLG
0270  LDD IBOT
0271  ADDD #$01
0272  CMPD #150
0273  BEQ ATTOP
0274  STD IBOT
0275  BRA SETFLG
0276 ATTOP LDD #$0000
0277  STD IBOT
0278 SETFLG LDD ITOP
0279  CMPD IBOT
0280  BLO WORKUP
0281  SUBD IBOT
0282  STD IFLAG
0283  BRA STOR
0284 WORKUP ADDD #150
0285  SUBD IBOT
0286  STD IFLAG
0287 STOR LDX TEMPX
0288  LDD IRAM1
0289  STD ,X++
0290  LDD IRAM2
```

```
0291  STD ,X++
0292  LDD IRAM3
0293  STD ,X++
0294  LDD IRAM4
0295  STD ,X++
0296  LDD IRAM5
0297  STD ,X++
0298  LDD IRAM6
0299  STD ,X++
0300  LDD IRAM7
0301  STD ,X++
0302  LDD IRAM8
0303  STD ,X++
0304  LDD IRAM9
0305  STD ,X++
0306  LDD IRAM10
0307  STD ,X++
0308  CMPX #$8CE6      ;HIGHEST ADDRESS OF DATA STACK
0309  BHS LOOPX
0310  STX TEMPX
0311  BRA DONE
0312 LOOPX LDX #$812E ;BOTTOM ADDRESS OF DATA STACK
0313  STX TEMPX
0314  LDD #$00
0315  STD ITOP
0316 DONE RTI
0317  END
```

```
0001 C       SUBROUTINE SEARCH
0002 C
0003 C
0004        SUBROUTINE SEARCH (LOS,JK,MASK)
0005 *
0006 ********************************** COMMON BLOCK ******************************
0007 *
0008              INCLUDE 'COMMON.SA'
0009 *
0010 ***************************************************************************
0011 *
0012 C            SEARCH FROM LOW TO HIGH FREQ. (SIGNAL MORE OFTEN LOW)
0013 *
0014        BND = SIGMIN
0015 *
0016 C            COUNTS ( KB,KBL,KBLL) BELOW MOVING BOUNDS,
0017 C            OBTAINS BAND COUNT ( KBNDG) AND INDEX ( IBND)
0018 *
0019        KBL= 0
0020        KBLL = 0
0021        KBNDG = 0
0022        IBND = 3
0023        DO 662   IB = 1,IN
0024        KB = 0
0025        DO 661 J= 1,10
0026   661  IF (FREQ(J).LT. BND ) KB = KB+1
0027        KBND = KB - KBLL
0028        BND = BND + HGATE
0029         IF (KBLL .EQ. 10) GO TO 10
0030        KBLL = KBL
0031        KBL = KB
0032        IF( IB .LT. 3 ) GO TO  662
0033        IF ( KBND .LT . KBNDG ) GO TO 662
0034        IBND = IB
0035        KBNDG = KBND
0036   662  CONTINUE
0037    10  IF(KBNDG .GE. 3) GO TO 664
0038 *
0039 C            SIGNAL NOT FOUND # INCREMENT NOISE COUNT
0040 *
0041        LOS = LOS +1
0042
0043        IF (LOS .GT. 100 ) GOTO 1111
0044        IF (LOS .EQ. 1) TSWCH2= TIME(1)
0045 C
0046 C            GET NEXT FIVE RAW DATA POINTS
0047 *
0048        RETURN
0049 C
0050 C
0051 C            FOUND SIGNAL # SET GATE , NOTE LEADING
0052 C            EDGE SWITCH TIME AND DWELL.
0053 C            CONDENSE THE DATA POINT , AND PROCEED TO DECOMMUTATE.
0054 *
0055   664  CONTINUE
0056        LOSN = LOS
0057        LOS = 0
0058        SIGLEV  =       SIGMIN  +  (IBND-2)*HGATE
```

```
0059 *
0060 *
0061        TSWCH1 = TSWTCH
0062        IF(TIME(1) .LT. TSWTCH)GOTO 44
0063         TSWTCH= TIME(1)
0064 44     DWELL = (TSWTCH - TSWCH1)
0065 C
0066 C           CONDENSER OUTPUT
0067 *
0068         IF(NSUM .EQ. 0) GO TO 2665
0069         IF(JK .EQ. 0)GOTO 4889
0070  7017   IF(TSWCH1 .LE. COND(1,JK))GO TO 3006
0071 4889   JK=JK+1
0072        COND(1,JK) = TSWCH1
0073        COND(2,JK) = DWELL
0074        COND(3,JK) = FSUM/NSUM
0075        IF(JK .GE. 900)GOTO 2222
0076 C
0077 C           DECOMMUTATE
0078 *
0079        IF (JK .GT. 1)GO TO 3001
0080        WRITE(100,3005)
0081  3005   FORMAT('****** COND. MATRIX ******',/)
0082 3001    CONTINUE
0083         WRITE(100,3000)JK,COND(1,JK),COND(2,JK),COND(3,JK)
0084  3000   FORMAT(' COND(',I3,')= ',F12.1,F8.1,F10.4,5X)
0085 C
0086         GO TO 2665
0087  1111   MASK = 1
0088         RETURN
0089  2222   MASK = 2
0090         RETURN
0091  3006 WRITE(IOUT,3007)JK
0092  3007 FORMAT(' *** WARNING ***  SWITCH TIME DID NOT INCREASE AT',
0093 &        ' JK = ',I3)
0094  2665 FSUM = 0.
0095        NSUM = 0
0096 C
0097        RETURN
0098         END
```

```
0001  OPT L,REL
0002  OPT LLE=80
0003  NAM SETUP
0004  IDNT SETUP
0005  XDEF SETUP
0006 *
0007 ********** THIS IS THE MAIN PAPER TAPE SETUP PROGRAM **********
0008 *
0009  PSCT
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 BYTE15 RMB 1
0026 BYTE16 RMB 1
0027 BYTE17 RMB 1
0028 BYTE18 RMB 1
0029 BYTE19 RMB 1
0030 BYTE20 RMB 1
0031 IRAM1 RMB 2
0032 IRAM2 RMB 2
0033 IRAM3 RMB 2
0034 IRAM4 RMB 2
0035 IRAM5 RMB 2
0036 IRAM6 RMB 2
0037 IRAM7 RMB 2
0038 IRAM8 RMB 2
0039 IRAM9 RMB 2
0040 IRAM10 RMB 2
0041 ITOP RMB 2
0042 IBOT 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
```

```
0059  STA BYTE9              ENTRY COUNTER
0060  STA BYTE10
0061  STA BYTE18
0062  STD IRAM1              STORAGE FOR INTEGER
0063  STD IRAM2              STORAGE FOR FRACTION
0064  STD IRAM3
0065  STD IRAM4
0066  STD IRAM5
0067  STD IRAM8
0068  STD IFLAG
0069  LDA #$04
0070  STA >$E017
0071  LDD #$8900
0072  STD IRAM6
0073  LDD #$5878         ;ADDRESS OF TAPE1
0074  STD >$DFFE         ;IRQ VECTOR JUMP
0075  LDA #$07
0076  STA >$E016
0077  ANDCC #$EF
0078  RTS
0079  END
0080
0081
```

```
0001  OPT L,REL
0002  OPT LLE=80
0003  NAM STOPER
0004  IDNT STOPER
0005  XDEF STOPER
0006 *
0007 *       THIS PROGRAM LOOKS TO SEE IF THE 'S' KEY OF THE TERMINIAL
0008 *      HAS BEEN HIT. IF SO THIS PROGRAM SET ISTPER TO 1. THEN ADVANCE
0009 *      LOOKS AT THIS FLAG, AND IF SET CAUSES THE PROGRAM TO STOP.
0010 *
0011  CSCT
0012 IFLAG RMB 2
0013 BYTE10 RMB 1
0014 BYTE9 RMB 1
0015 BYTE8 RMB 1
0016 BYTE7 RMB 1
0017 BYTE6 RMB 1
0018 BYTE5 RMB 1
0019 BYTE4 RMB 1
0020 BYTE3 RMB 1
0021 BYTE2 RMB 1
0022 BYTE1 RMB 1
0023 BYTE11 RMB 1
0024 BYTE12 RMB 1
0025 BYTE13 RMB 1
0026 BYTE14 RMB 1
0027 ISTPER RMB 2
0028 TEMPX RMB 2
0029 INTMSK RMB 2
0030 IRAM1 RMB 2
0031 IRAM2 RMB 2
0032 IRAM3 RMB 2
0033 IRAM4 RMB 2
0034 IRAM5 RMB 2
0035 IRAM6 RMB 2
0036 IRAM7 RMB 2
0037 IRAM8 RMB 2
0038 IRAM9 RMB 2
0039 IRAM10 RMB 2
0040 ITOP RMB 2
0041 IBOT RMB 2
0042 IDATA RMB 1500    ;IRAM ARE STACKED INTO THE IDATA ARRY TO BE PROCESSED
0043 **************************
0044 *
0045 *
0046 *  PROGRAM FOLLOWS
0047 *
0048  PSCT
0049 STOPER LDD #$0000
0050  STD ISTPER
0051  LDA >$EC14
0052  LSRA
0053  BCC NOCHAR      ;IF FLAG NOT SET THEN NO CHAR
0054  LDA >$EC15      ;LOAD THE DATA FROM THE ACIA
0055  CMPA #$53       ;ASCII LETTER S
0056  BNE NOCHAR      ;IF NOT EQUAL TO S RETURN
0057  LDD #$0001      ;TO SET THE FLAG
0058  STD ISTPER      ;FLAG IS NOW SET
```

```
0039 NOCHAR RTS
0040  END
```

```
0001  OPT L,REL
0002  OPT LLE=80
0003  NAM TAPE1
0004  IDNT 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 INTERRUP. 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 *       TABLE.
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 IBOT 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 IRAM4
0056  LDY IRAM6
0057  LDA >$E014
0058  LDA >$E015   LOAD DATA INTO ACC.
```

```
0059  STA BYTE2
0060  LDB BYTE7    THREE CONSECUTIVE DELETES?
0061  CMPB #$05
0062  BEQ SPAC
0063  CMPA #$7F
0064  BEQ RED
0065  LDB #$00
0066  STB BYTE7       RESET DELETE FLAG
0067  BRA OUT
0068 RED INC BYTE7
0069 OUT STA BYTE2
0070  STY IRAM6
0071  LDY IRAM4
0072  RTI
0073 SPAC CMPA #$20
0074  BNE NSPAC
0075  LDB #$06
0076  STB BYTE8            COUNTER SET TO SIX
0077 NSPAC CMPA #$2E
0078  BNE NPER
0079  LDB #$01
0080  STB BYTE1           SET D.P. FLAG
0081  BRA OUT
0082 NPER CMPA #$30
0083  BLO OUT              NUMBER?
0084  CMPA #$39
0085  BLS MASK
0086  BRA OUT
0087 MASK ANDA #$0F
0088  STA ,-Y         STORE ON STACK
0089  LDB BYTE8
0090  DECB
0091  STB BYTE8
0092  LDA BYTE18      120 ENTRIES?
0093  CMPA #120
0094  BLO NZER        ACCEPTS ZEROS AFTER 160 ENTRIES.
0095  LDA BYTE2
0096  ANDA #$0F
0097  CMPA #$00
0098  BNE NZER
0099  LDB BYTE8
0100  CMPB #$05       FIRST DIGIT?(ACCEPTING ZEROS)
0101  BEQ ZER
0102 NZER LDB BYTE1
0103  CMPB #$01      CHECK D.P. FLAG
0104  BEQ RES
0105  BRA OUT
0106 RES LDB #$00
0107  STB BYTE1      RESET D.P. FLAG
0108 ZER LDD #$0000
0109  STD IRAM1      INITIALIZE TO ZERO
0110  STD IRAM2
0111  LDB ,Y+        .X=D ACCUMULATOR
0112  INC BYTE8
0113  STD IRAM2
0114  LDB #$01
0115 MULT STB BYTE3 MULTERPLIER = X
0116  LDA BYTE8
```

```
0117  CMPA #$05      LAST DIGIT?
0118  BEQ LAST
0119  CMPA #$06      FINISHED WITH ENTRY?
0120  BEQ FINI
0121  CMPA #$02      1000'S PLACE?
0122  BNE THOU
0123  LDA BYTE8
0124  INC BYTE6      SET 1000'S FLAG
0125 THOU LDA ,Y+
0126  INC BYTE8
0127  MUL            MULTIPLY BY POWER OF TEN
0128  ADDD IRAM1
0129  STD IRAM1
0130  LDA BYTE3
0131  LDB #10
0132  MUL            INCREMENT MULTIPLIER
0133  BRA MULT
0134 LAST LDA BYTE6
0135  CMPA #$01
0136  BNE THOU
0137  DEC BYTE6      RESET 1000'S FLAG
0138  LDA ,Y+
0139  INC BYTE8
0140  LDB #$0A       TEN
0141  MUL            TIMES BY TEN
0142  LDA #$64       100
0143  MUL            TIMES BY 100
0144  ADDD IRAM1
0145  STD IRAM1
0146 FINI LDD #$0001    SET FLAG
0147  STD IFLAG
0148  INC BYTE18
0149  LDA #$06
0150  STA BYTE8      RESET COUNTER
0151  JMP OUT
0152  END
```

```
0001 C      TRACK SUBROUTINE
0002       SUBROUTINE TRACK(LOS,JK,MASK)
0003 *
0004 ********************************* COMMON BLOCK**********************************
0005 *
0006             INCLUDE 'COMMON.SA'
0007 *
0008 ********************************************************************************
0009 C
0010 C           SET GATE BOUNDS
0011 C           COUNT SIGNAL POINTS IN GATE
0012 *
0013       MASK=0
0014       BUPR = SIGLEV + HGATE
0015       IF(BUPR.GT.SIGMAX ) BUPR = SIGMAX
0016       BLWR = SIGLEV - HGATE
0017       IF(BLWR.LT.SIGMIN ) BLWR = SIGMIN
0018 *
0019 C           MEMORY TO STABILIZE CONDENSED SIGNAL
0020 *
0021       NGATE = 1
0022       SUMGTE = SIGLEV
0023 *
0024 C           COUNT POINTS  IN GATE
0025 *
0026       DO 671  J= 1,10
0027       IF (FREQ(J) .GT. BUPR .OR. FREQ(J) .LT. BLWR) GOTO 671
0028       SUMGTE = SUMGTE + FREQ(J)
0029       NGATE = NGATE+1
0030       IF ( NGATE .GT. 6) GO TO 672
0031   671 CONTINUE
0032 *
0033 C           IF LESS THAN TWO (EXCLUDING SIGLEV) IN GATE,  LOST SIGNAL
0034 *
0035       IF ( NGATE .GT. 2 ) GO TO 672
0036        CALL SEARCH(LOS,JK,MASK)
0037        IF (MASK .EQ. 1) GOTO 83
0038        IF (MASK .EQ. 2) GOTO 85
0039        RETURN
0040 *
0041 C           HAVE SIGNAL # INCREMENT FOR MEAN AND ADJUST GATE
0042 *
0043   672 FSUM = FSUM + SUMGTE/NGATE
0044       NSUM = NSUM +1
0045       SIGLEV = (SIGLEV + SUMGTE / NGATE) * 0.5
0046       RETURN
0047 C
0048  83    CONTINUE
0049 C
0050 C     LOST SIGNAL --- NEVER FOUND IT IN SEARCH
0051 C
0052        MASK=1
0053        RETURN
0054 C
0055  85       CONTINUE
0056 *
0057 C                  JK > 1000  (COND DIMENSION)
0058 *
```

```
0059          MASK=2
0060          RETURN
0061
0062        END
```

```
0001 CADIR    FUNCTION CALLED BY WINDS
0002 C
0003       FUNCTION ADIR(U,V)
0004 C
0005 C  THIS SUBROUTINE WILL COMPUTE THE DIRECTION FROM WHICH THE WIND IS BLOWIN
0006 C        THIS WILL NEED DIMENSION IFF ACTUAL ARGUMENTS ARE SUBSCRIPTED
0007 C
0008         DUMMY=V/U
0009   60    IF(U)61,62,63
0010   61   ADIR = 90. - 57.2958* ATAN(DUMMY)
0011          GO TO 64
0012   63   ADIR = 270. - 57.2958* ATAN(DUMMY)
0013          GO TO 64
0014   62    IF (V) 65,66,67
0015   65   ADIR = 360.
0016         GO TO 64
0017   66   ADIR = 999.
0018         GO TO 64
0019   67     ADIR = 180.
0020   64    RETUR
0021       END
```

```
0001 C FUNCTION TO TAKE MOD OF REAL #
0002   FUNCTION AMOD(A,B)
0003 **
0004 **
0005       DUMMY=A/B
0006       IDUM=DUMMY
0007       DUMMY2=IDUM
0008       AMOD=(DUMMY-DUMMY2)*B
0009       RETURN
0010       END
```

```
0001 C  FUNCTION TO TAKE BASE 10 LOG OF A NUMBER
0002    FUNCTION ALOG10(A)
0003 *
0004 *
0005      ALOG10=(ALOG(A))/(ALOG(10.))
0006      RETURN
0007      END
```

```
0001 C  FUNCTION TO TURN INTEGER TO REAL NUMBER
0002 *
0003    FUNCTION FLOAT(I)
0004 *
0005   DUMMY=I
0006   FLOAT=DUMMY
0007   RETURN
0008   END
```

```
0001 C              ROUTINE FOR  REDUCTION OF ONE MINUTE DATA
0002 C
0003        SUBROUTINE PRD
0004 ***************************************************************************
0005 *
0006        INCLUDE 'PRDCOM.SA'
0007 *
0008 ***************************************************************************
0009        DIMENSION IIERR(8),OLD(5),IERRM(488),AKKMB(36) ;TEST IERRM&AKKMB
0010        DIMENSION EPR(110),ETP(110),EPT(110),NLBL(6) ;NLBLFOR TEST
0011        DIMENSION P(110),P3(110),TEMP(110),DLBL(2)  ;DLBL FOR TEST
0012 C
0013        DATA BI/0.0/
0014        DATA AKKMB/
0015 &         1000.0,900.0,850.0,800.0,700.0,600.0,500.0,400.0,350.0,300.0,
0016 &         250.0,200.0,150.0,125.0,100.0,80.0,70.0,60.0,50.0,40.0,35.0,
0017 &         30.0,25.0,20.0,17.5,15.0,12.5,10.0,8.0,7.0,6.0,5.0,4.0,3.5,
0018 &         3.0,1.0/
0019 ***************************************************************************
0020 C                    CONSTANTS USED IN PROGRAM
0021 C
0022        DATA ZERO/0.0/,IZERO/0/,ONE/1.0/,IONE/1/,TEN/10.0/    ;TESTING
0023        DATA ISEVEN/7/,SEVEN/7.0/,ZNINE/-99.99/,Z180/180.0/   ;TESTING
0024        DATA ZNN/-999.99/,TWO/2.0/,POINT5/0.5/,ZTENP2/10.2/    ;TESTING
0025        DATA Z99/99.0/,Z98P9/98.9/,Z56/56.0/,Z113/113.0/,Z57/57.295779/
0026        DATA POINT1/0.1/,Z99P9/99.9/,Z9P9X5/9.99999/,Z99P99/99.99/
0027        DATA Z999P9/999.9/,ZNNN/-999.999/,ZP9966/.996626/,Z48000/48000.0/
0028        DATA Z8P9E/8.902E6/,Z294P/294.0/,Z4P3E/4.3E-3/,ZP611/.611/
0029        DATA Z1P7E/1.74533E-2/,Z14P6/14.64467/,Z3P9E/3.9453635E-4/
0030        DATA Z7P8E/7.89688E-4/,Z20P/20.0/,Z273P1/273.15/,Z233P1/233.15/
0031        DATA Z3P29E/3.298588E-3/,Z1P07E/1.07701E-3/,Z5P89E/5.89986E-5/
0032        DATA RATE/0.0/
0033 C
0034 ***************************************************************************
0035 *
0036 *
0037   100 FORMAT(1X,2A5,2I7,5F6.1,I2,2F5.1,F7.1,A6)
0038   101 FORMAT(F6.1,I6,F6.1,F8.5,F7.1,F6.2,F7.1,F7.4,F6.1,F7.1,F7.2,2F6.1,
0039 &          F7.4,4F6.1,F6.1,10X)
0040  1010 FORMAT(F6.1,2X,F6.0,2X,F7.1,2X,F7.4,2X,F6.1,2X,F7.1,2X,F7.2,2X,F6.1,
0041 &         2X,F6.1,2X,F7.4,2X,4(F6.1,2X))
0042   109 FORMAT('0TIME     ALT      PRESS      ',
0043 &    'LOG      TEMP     PTEMP     VTEMP     HUMTY    DEWPT    SPECIF      ',
0044 &    'SPD      DIR      NS       EW',
0045 &    /,'MIN      GP MT      MB        PRESS     DEG K    DEG K     ',
0046 &    'DEG K     FRCNT    DEG K    HUMTY       MPS      DEG      MPS     MPS',/)
0047   110 FORMAT(5X,'SURFACE CONDITIONS',/,
0048 &       7X,'PRESS',F8.1,' MB',8X,'BASE CAL = ',F5.1,' C AT ',F5.1,' ORD',
0049 &       /,7X,'TEMP',F9.1,' K',9X,'HUMIDITY = ',F5.1,' % AT ',F5.1,' ORD',
0050 &       /,7X,'HUMY',F9.1,' %',//,120('*'))
0051  120 FORMAT('1STATION',I4,4X,'LAUNCH DATE',I3,'-',I2,'-',I2,4X,
0052 &          'LAUNCH TIME',I2,':',I2,':',F3.1,' GMT    SONDE ID ',I5,/,/)
0053   152 FORMAT(2X,I3,1X,3I4,62X)
0054   153 FORMAT(2I7,I5,F7.1,3F5.0,F4.0,35X)
0055   210 FORMAT(2A5,I6,A5,'ECC  ',A6,F8.1,4F6.1,3F5.1,F5.3,F5.1,F7.1,F5.1,
0056 &         F6.1,F5.1,F7.2,F6.3,6X)
0057  1000 FORMAT(1X,A6,A4,1X,A6,A5,2X,2F6.1,F6.2,3F5.1,F6.2,F5.3,5X,A6)
0058  4000 FORMAT(/,'*** RECORDED INSTRUMENT HUMIDITIES ***',/,
```

```
0059 &     '*** LESS THAN 20 POINT NOT LISTED  ***')
0060  5000  FORMAT(/,'***     GMD IN LIMITING ANGLES     ***',/,
0061 &           '***   DURING PART OF OBSERVATION   ***')
0062  6011  FORMAT(5F12.5,A10,2X,I2,A29,29X)
0063  8150 FORMAT(/,'****** ORDINATE GREATER THAN 100.0 ******'/
0064 &       F5.0,F7.1,2F4.1,F7.2,F6.2,2X)
0065  8160 FORMAT(/,'******    GMD TRACKING ERROR       ***'/
0066 &       F5.0,F7.1,2F4.1,F7.2,F6.2,2X)
0067  9034 FORMAT('   ***    ',A61)
0068  9036 FORMAT('+',71X,F10.4,'   **********')
0069  9154 FORMAT(/)
0070  9264  FORMAT(/,'*** LAYER BELOW ',F6.1,' MB HAS SUPER ADIABATIC LAPSE'
0071 &     ,' RATE OF ',F5.1,' DEG/KM')
0072  9364 FORMAT(/,'*** LEVEL BELOW ',F6.1,' MB   ***   POTENTIAL ',
0073 &      'TEMPERATURE = ',F6.1,' DEG K   ***   NOT INCREASING')
0074  9464  FORMAT(/,'******    FOR THE ABOVE LAYERS OR LEVELS, CHECK TEMP ',
0075 &           'ORDINATE AND PRESSURE ENTRIES.    ******',/,/)
0076 C
0077 C
0078 C
0079 C     READ RADIOSONDE CALIBRATION CARD
0080 C
0081       IOUT=102
0082       HTMSL=HEIGHT
0083       CALTP=30.0
0084       RECRH=46.0
0085 C
0086 C
0087 C     SET INITIAL FLAGS
0088 C
0089        AA0=0.00191617
0090        AA1=0.0173393
0091        AA2=-0.2103E-04
0092        AA3=0.851677E-07
0093        AA4=-0.181232E-09
0094        AA5=0.135529E-12
0095       IH=0
0096       IXT=0
0097       ITEMP=0
0098       IERRF=0
0099       MIN=1
0100       KKKK=1
0101       ICALFL=1
0102       IHUMFL=1
0103       IELVFL=0
0104       IOZNFL=0
0105       CENT=3.16
0106       OI=OIZ
0107       C1=OIC
0108       TBI=TBO
0109       PRESS=PCAL + .05
0110       IF(A1 .NE. 0) GO TO  555
0111       A0=0
0112       A1=1
0113       ICALFL=0
0114  555  IOTME=-999
0115       IF(KOPT .EQ. 0) ICALFL=0
0116       NLBL(5)='6H'
```

```
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 TEMPCE(CALTP,IERRF,I,ITEMP)
0126       CALL RL(RECRH,CALTP,I,IDF,IH)
0127 C
0128 C  READ HEADER CARD
0129 C
0130      IF(NCDS .GE. 108)NCDS = 108
0131 C      **** PROCOMMON ALLOWS ONLY 108 ONE MINUTE ENTRIES
0132 C      **** WHERE COND PASSES UP TO 110 MINUTES OF DATA
0133       IZR=0
0134       IBRST=NCDS-1
0135       ITERM=NCDS-1
0136       DUMMY=PR(NCS+1)
0137       DUMMY = ONE
0138 C
0139 C  READ DATA CARD
0140 C
0141       DO 577 I=1,NCDS
0142 *
0143 *     DATA IS PASSED FROM MET TO PRD IN THE VL ARRAY
0144 *
0145       TIM(I)=VL(1,I)/60.0
0146       PR(I)=VL(4,I)
0147       DT(I)=VL(6,I)
0148       DF(I)=VL(7,I)
0149       AZ(I)=VL(2,I)
0150       EL(I)=VL(3,I)
0151 577  CONTINUE
0152 C
0153       HGMDT=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       DEWPT(I)=ZERO
0162       IF(TIM(I) .LE. IOTME) IIERR(2)=1
0163       IF(I .GT. NCDS) IIERR(3)=1
0164       IOTME=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)=ZNINE
0169       EL(I)=ZNINE
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)+Z180
0176     14 IF(PR(I) .NE. ZERO) 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(1)=PR(I)
0181        IIERR(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 9189 JJ=1,IXT
0188        K=JJ+ILO
0189        DUMMY=JJ
0190        DUMMY1=IHI-ILO
0191        TALP(K)=TALP(ILO)+DUMMY/DUMMY1*(TALP(IHI)-TALP(ILO))
0192   9189 DUMMY = TALP(K)
0193        PR(K)=POWER(TEN,DUMMY)
0194        IXT=0
0195     18   IF(I .GE. NOBS-1) GO TO 36
0196        IF(DT(I).EQ.ZERO.OR.DT(I+1).EQ.ZERO.OR.DT(I+2).EQ.ZERO) GO TO 36
0197         DUMMY=DT(I)-DT(I+1)
0198         DUMMY1=DT(I)-DT(I+2)
0199         D1=ABS(DUMMY)
0200         D2=ABS(DUMMY1)
0201         IF(D1 .GE.POINT5 .AND. D1 .GE. D2*TWO) IIERR(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(IIERR(II) .EQ. 0) GO TO 38
0209        IIERR(II)=0
0210        IF(II .GT.1)GOTO 2224
0211        WRITE(IOUT,1115)
0212   1115 FORMAT('   DATA CARD ID DOES NOT MATCH FLIGHT ID ***')
0213   2224 IF(II .GT. 2)GOTO 3334
0214        WRITE(IOUT,2225)
0215   2225 FORMAT(' TIME NOT INCREASING - PROGRAM CONTINUES ***')
0216   3334 IF(II .GT. 3) GOTO 4444
0217        WRITE(IOUT,3335)
0218   3335 FORMAT('TIME GREATER THAN 120 MIN - PROGRAM CONTINUES')
0219   4444 IF(II .GT. 4) GOTO 5554
0220        WRITE(IOUT,4445)OLD(II-3)
0221   4445 FORMAT('PRESSURE NOT DECREASING - PRESSURE=',F10.4)
0222   5554 IF(II .LT. 5) GOTO 38
0223       WRITE(IOUT,5555)OLD(II-3)
0224   5555 FORMAT('TEMP ORD FAILS DIFFERENCE TEST ** CHECK TEMP ORD AT ',F10.4)
0225     38 CONTINUE
0226 C
0227        DUMMY=100
0228        IF (DT(I) .GT.DUMMY .OR. OF(I) .GT.DUMMY)
0229 &         WRITE(IOUT,8150) TIM(I),PR(I),
0230 &         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),
```

```
0233 &         DT(I),DF(I),AZ(I),EL(I)
0234     40 CONTINUE
0235 C
0236 C
0237 C  PROCESS DATA POINTS
0238 C
0239        DO 90 I=1,NCDS
0240        PE=ONE
0241        X=PR(I)
0242        J=I
0243 C
0244 C      RADIOSONDE TEMPERATURE CONVERSION
0245 C
0246 *
0247        CALL TEMPCE(CALTP,IERRF,J,ITEMP)
0248 C
0249 C      RADIOSONDE RELATIVE HUMIDITY CONVERSION
0250 C
0251 *
0252        CALL RL(RECRH,CALTP,J,IDF,IH)
0253 C
0254        DUMMY=(1000)/PR(I)
0255        DUMMY1=POWER(DUMMY,.28571428)
0256        POTTP(I)=T(I)*DUMMY1
0257 ************************************************************
0258 C
0259 C      COMPUTE ALTITUDE
0260 C
0261        IF (I .GT. 1) GO TO 62
0262        DUMMY=PCAL/PR(I)
0263        HGP(I)=HTMSL+Z14P6*ALOG(DUMMY)*(T(I)+SURT+CENT)
0264        GO TO 11
0265    62  DUMMY=PRIMEP/PR(I)
0266        DLOGP=ALOG(DUMMY)
0267        HGP(I)=PRIMEH+Z14P6*DLOGP*(TV(I)+PRIMEV)
0268    11  PRIMEP=PR(I)
0269        PRIMEH=HGP(I)
0270        PRIMEV=TV(I)
0271        IF (DF(I) .EQ. ZERO) DF(I)=Z99P9
0272         IF(IDF .NE. 1) GO TO 83
0273         F(I)=BI
0274         DEWPT(I)=BI
0275    83   CONTINUE
0276 C
0277 C       CHECK LAPSE RATE AND POTENTIAL TEMPERATURE
0278 C
0279         EPR(I)=ZNN
0280         ETP(I)=ZNN
0281         EPT(I)=ZNN
0282         IF(DT(I).EQ.ZERO) GO TO 90
0283         IF(I .EQ. 1) GO TO 90
0284         DUMMY=HGP(I-1)-HGP(I)
0285         RATE=((T(I-1)-T(I))*(TEN**3))/ABS(DUMMY)
0286         IF(RATE .LT. ZTENP2) GO TO 84
0287         ETP(I)=RATE
0288         EPR(I)=PR(I)
0289    84   IF(POTTP(I) .GT. POTTP(I-1)) GO TO 90
0290         EPT(I)=POTTP(I)
```

```
0291          EPR(I)=PR(I)
0292 C
0293    90    CONTINUE
0294 C
0295 C
0296          SURTK=SURT+CENT
0297          IF(IOZNFL.EQ.0.AND.PR(NCDS).LE.Z20P) GO TO 700
0298          SUMR=BI
0299          O3TOT=BI
0300    700   ISN=AMOD(LANCH,100)
0301          WRITE(IOUT,120)IDSTAT,IMONTH,IDAY,IYEAR,I1,I2,TS3,IDSOND
0302          WRITE(IOUT,110)PCAL,CALTP,RECTP,SURTK,CALRH,RECRH,SURH
0303          WRITE(IOUT,109)
0304          IF(SUMR .EQ. BI) SUMR=Z9P9X5      ;Z9P9X5=9.99999
0305 C
0306 C       WINDS COMPUTATION
0307 C
0308          MAX=NCDS
0309    301   CONTINUE
0310          IF(PR(1) .GT. AKKMB[illegible] GO TO 311
0311          KKKK=KK[illegible]
0312          [illegible]
0313    3[illegible]      ;Z9P9=999.9
              [illegible]9
[illegible]
              DO 94 I=MIN,MA[illegible]
              SPD([illegible]
              DIR[illegible]
              [illegible]
[illegible]
              [illegible]
[illegible]
              [illegible]
              IM1=I-1
              IP1=I+1
              [illegible]
              IF I .GT. 1 .AND. I .LT. MAX GO TO 863
              IM1 .EQ. MAX GO TO 863
              [illegible]
              DIR[illegible]
              GO TO 763
[illegible]   DUMMY=-Z99P99       ;Z99P99=99.99
0333          IF(AZ(IM1) .EQ. DUMMY .OR.  AZ(IP1) .EQ. DUMMY) GO TO 863
0334    763   ANG=(DIR(I)-Z180)*Z1P7E
0335          WNS=SPD(I)*ZCOS(ANG)    ;6809 CANT DO NEG # SO WRITE FUNCTION TO
0336          WEW=SPD(I)*ZSIN(ANG)    ;                    HANDLE IT
0337          GO TO 9663
0338    863   DIR(I)=Z999P9     ;Z999P9=999.9
0339          SPD(I)=Z999P9
0340          WNS=Z999P9
0341          WEW=Z999P9
0342   9663   QQ=(TV(I)/T(I)-ONE)/ZP611
0343          IF(F(I) .EQ. BI) QQ=BI
0344          P(I)=PR(I)
0345          TEMP(I)=T(I)-Z273P1
0346 C
0347 C            INTERPOLATE WIND VALUES
0348 C
```

```
0349        IF(WNS .NE.Z999P9 .AND. WEW .NE.Z999P9) GO TO 8863
0350        IF(I .EQ. MAX .OR. I .EQ. 2)   GO TO 8863
0351        IF(AZ(IM2) .EQ. ZNINE) GO TO 8863
0352        ANG=(DIR(IM1)-Z180)*Z1P7E
0353        WNS88=SPD(IM1)*ZCOS(ANG)
0354        WEW88=SPD(IM1)*ZSIN(ANG)
0355        ANG=(DIR(IP1)-Z180)*Z1P7E
0356        WNS99=SPD(IP1)*ZCOS(ANG)
0357        WEW99=SPD(IP1)*ZSIN(ANG)
0358        WNS=WNS88-((WNS88-WNS99)/TWO)
0359        WEW=WEW88-((WEW88-WEW99)/TWO)
0360  8863  IF(I .GT. 1) GO TO 610 .
0361        WEWLST=WEW
0362        WNSLST=WNS
0363 C
0364 C           INTERPOLATE STANDARD MET ALTITUDES
0365 C
0366   610  IF(PR(I) .GE. AKKMB(KKKK)) GO TO 9963
0367        DUMMY=PR(IM1)
0368        DUMM1=PR(I)
0369        DUMM2=AKKMB(KKKK)
0370        A=(ALOG10(DUMMY)-ALOG10(DUMM2))
0371        B =(ALOG10(DUMMY)-ALOG10(DUMM1))
0372        C=A/B
0373        AA99=TIM(IM1)+C*(TIM(I)-TIM(IM1))
0374         BB99=HGP(IM1)+C*(HGP(I)-HGP(IM1))
0375         CC99=BI
0376         DD99=BI
0377         EE99=BI
0378         RR99=BI
0379  8862   CONTINUE
0380        FF99=AKKMB(KKKK)
0381        GG99=T(IM1)+C*(T(I)-T(IM1))
0382        HH99=P(IM1)+C*(P(I)-P(IM1))
0383        DD99=DEWPT(IM1)+C*(DEWPT(I)-DEWPT(IM1))
0384        IF(P(I+1) .NE. BI .AND. P(I) .NE. BI) GO TO 8864
0385         JJ99=BI
0386         QQ99=BI
0387  8864  PP99=TALP(IM1)+C*(TALP(I)-TALP(IM1))
0388        QQ99=POTTP(IM1)+C*(POTTP(I)-POTTP(IM1))
0389         SS99=Z999P9
0390         TT99=Z999P9
0391         UU99=Z999P9
0392         VV99=Z999P9
0393        IF(I .EQ. MAX .OR. I .EQ. 2) GO TO 6663
0394         IF(I .EQ. 1 .AND. WNS .NE.Z999P9) GO TO 866
0395        IF(AZ(IM2) .EQ. ZNINE .OR. AZ(IM1) .EQ. ZNINE .OR.
0396 &         AZ(I) .EQ. ZNINE .OR. AZ(IP1) .EQ. ZNINE) GO TO 6663
0397    866 ANG=(DIR(IM1)-Z180)*Z1P7E
0398        WNS99=SPD(IM1)*ZCOS(ANG)
0399        WEW99=SPD(IM1)*ZSIN(ANG)
0400        UU99=WNS99+C*(WNS-WNS99)
0401        VV99=WEW99+C*(WEW-WEW99)
0402         DUMMY=VV99*VV99+UU99*UU99
0403         SS99=SQRT(DUMMY)
0404         DUMMY=VV99/UU99
0405         TT99=ATAN(DUMMY)*Z57
0406         IF(VV99 .LT.ZERO .AND. UU99 .GT.ZERO) TT99=TT99+Z180
```

```
0407          IF(VV99 .GT.ZERO .AND. UU99 .GT.ZERO) TT99=TT99+Z1800
0408          IF(VV99 .GT.ZERO .AND. UU99 .LT.ZERO) TT99=TT99+(2*Z180)
0409   6663 WW99=TV(IM1)+C*(TV(I)-TV(IM1))
0410         XX99=(WW99/GG99-ONE)/ZP611
0411         XX99=XX99+C*(QQ-XX99)
0412         IF(WW99 .EQ. BI) XX99=BI
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(WEW .EQ.Z999P9 .OR. WNS .EQ.Z999P9) GO TO 600
0417         UU99=WNSLST+C*(WNS-WNSLST)
0418         VV99=WEWLST+C*(WEW-WEWLST)
0419          DUMMY=VV99*VV99+UU99*UU99
0420          SS99=SQRT(DUMMY)
0421          DUMMY=VV99/UU99
0422          TT99=ATAN(DUMMY)*Z57
0423          IF(VV99 .LT.ZERO .AND. UU99 .GT.ZERO) TT99=TT99+Z180
0424          IF(VV99 .GT.ZERO .AND. UU99 .GT.ZERO) TT99=TT99+Z180
0425          IF(VV99 .GT ZERO .AND. UU99 .LT.ZERO) TT99=TT99+(2*Z180)
0426    600 IF(T(I).LE.Z233P1 .OR. F(I).EQ.BI .OR. F(I).GT.Z20P) GO TO 900
0427         IHUMFL=0
0428         F(I)=BI
0429         DEWPT(I)=BI
0430         QQ=BI
0431         IF(GG99.LE.Z233P1 .OR. HH99.EQ.BI .OR. HH99.GT.Z20P) GO TO 900
0432         HH99=BI
0433         OO99=BI
0434         XX99=BI
0435    900 IF(TV(I) .NE. ZNN)GO TO 800
0436         T(I)=0.
0437         TV(I)=0.
0438         POTTP(I)=0.
0439         HGP(I)=0.
0440         GG99=0.
0441         QQ99=0.
0442         DD99=0.
0443         BB99=0.
0444         WW99=0.
0445 C
0446 C              WRITE INTERPOLATIONS
0447 C
0448 800 AA99P = AA99 + .05
0449     BB99P = BB99 + .5
0450     FF99P = FF99 + .05      ;ROUNDED FOR PRINTING
0451     PP99P = PP99 + .00005
0452     GG99P = GG99 + .05
0453     QQ99P = QQ99 + .05
0454     WW99P = WW99 + .005
0455     HH99P = HH99 + .05
0456     OO99P = OO99 + .05
0457     XX99P = XX99 + .00005
0458     SS99P = SS99 + .05
0459     TT99P = TT99 + .05
0460     UU99P = UU99 +.05
0461     VV99P = VV99 + .05
0462     WRITE(IOUT,1010)AA99P,BB99P,FF99P,PP99P,GG99P,QQ99P,WW99P,HH99P,
0463    &          OO99P,XX99P,SS99P,TT99P,UU99P,VV99P
0464       KKKK=KKKK+1
```

```
0465  9963 IF(PR(I) .EQ.AKKME(KKKK)) KKKK=KKKK+1
0466       IF(T(I).LE.Z233P1 .OR. F(I).EQ.BI .OR. F(I).GT.Z20P) GO TO 910
0467       IHUMFL=0
0468       F(I)=BI
0469       DEWPT(I)=BI
0470       QQ=BI
0471   910 IF(TV(I) .NE. ZNN) GO TO 810      ;ZNN=-999.9
0472       T(I)=BI
0473       TV(I)=BI
0474       POTTP(I)=BI
0475       HGP(I)=BI
0476   810 CONTINUE
0477        IF(SPD(I) .NE.Z999P9 .AND. DIR(I) .NE.Z999P9) GO TO 811
0478        WNS=Z999P9
0479        WEW=Z999P9
0480   811  CONTINUE
0481 C
0482 C         WRITE PROCESSED DATA
0483 *
0484       TIMP = TIM(I) + .05  ;DUMMY VARIABLE FOR PRINTING IE ROUNDS NUMBER
0485       HGPP = HGP(I) + .5
0486       PRP = PR(I) + .05
0487       TALPP = TALP(I) +.00005
0488       TPPP = T(I) + .05
0489       POTTPP = POTTP(I) +.05
0490       TVP = TV(I) + .005
0491       FPPP = F(I) +.05
0492       DEWPTP = DEWPT(I) +.05
0493       QQP = QQ + .00005
0494       SPDP = SPD(I) + .05
0495       DIRP = DIR(I) + .05
0496       WNSP = WNS + .05
0497       WEWP = WEW + .05
0498        WRITE(IOUT,1010) TIMP,HGPP,
0499 &           PRP,TALPP,TPPP,POTTPP,TVP,FPPP,DEWPTP,QQP,SPDP,
0500 &           DIRP,WNSP,WEWP
0501    63 CONTINUE
0502 C
0503 C         WRITE MESSAGES
0504 C
0505        WRITE(IOUT,9154)
0506       IF(IHUMFL .EQ. 0) WRITE(IOUT,4000)
0507        IF(IELVFL .EQ. 1) WRITE(IOUT,5000)
0508        WRITE(IOUT,9154)
0509        IPTFLG=-1
0510        DO 64 I=1,NCDS
0511        IF(EPR(I) .LT. ZERO) GO TO 64
0512        IPTFLG=1
0513        IF(ETP(I) .GE. -999.) WRITE(IOUT,9264) EPR(I),ETP(I)  ;ZNNN=-999.999
0514        IF(EPT(I) .GE. -999.) WRITE(IOUT,9364) EPR(I),EPT(I)
0515   64   CONTINUE
0516        IF(IPTFLG .GT. 0) WRITE(IOUT,9464)
0517 C
0518 C
0519 C
0520    1   RETURN
0521       END
```

```
********** COMMON DATA FOR PRD PROGRAM **********
*
      COMMON TALP(100),AZ(100),DEWPT(100),DIR(100),EL(100),F(100)
      COMMON HGP(100),JUNK(123),VL(7,110),SECS,DUMMY,DUMMY1,IDUMM
      COMMON IDUMM1,AMINU,IDSTAT,IDSOND,IMONTH,IDAY,IYEAR,HEIGHT
      COMMON FP0,FTEMP0,FR0,RECTP,CALRH,ICREN,CERST,NCDS,IOUT
      COMMON VSFC,DSFC
      COMMON SPD(100),T(100),TIM(100),TV(100),XX(100)
      COMMON YY(100),DT(100),DF(100),PR(100)
      COMMON POTTP(100)
C
```

```
0001      OPTION STACK=300,USTACK=200
0002      PROGRAM MAIN
0003 C    'PRD' RADIOSONDE DATA REDUCTION PROGRAM
0004 C
0005 ********** COMMON DATA FOR PRD PROGRAM **********
0006 *
0007        INCLUDE 'PRDCOM.SA'
0008 *
0009 ************************************************************
0010 C
0011 C---DATA IS PASSED TO PRD FROM MET IN COMMONS
0012 C
0013        WRITE(IOUT,140)
0014 140    FORMAT(/,/,/,'********** OUTPUT PROCESSING FOLLOWS **********',/)
0015         WRITE(100,1000)
0016  1000   FORMAT('  AT THE START OF THE  PRD PROGRAM')
0017 *
0018 *
0019 *
0020         CALL PRD
0021 C
0022 ***** PROGRAM ENDS **********
0023 C
0024    20    WRITE(IOUT,130)
0025 130      FORMAT(' NORMAL PROGRAM TERMINATION')
0026        WRITE(IOUT,145)
0027 145    FORMAT(/,/,/,/,)
0028        WRITE(100,146)
0029 146    FORMAT(/,/,/,/,/,79('*'),/,/,/,/,/,37(' '),'UMET-1',/,
0030 &              27(' '),'NORMAL PROGRAM TERMINATION',
0031 &              /,/,/,/,/,79('*'),/,/,/,/,/,/,/,/,/)
0032 *
0033 *
0034        IDUM = 1        ;SET UP INFINTE DO LOOP
0035  300   IDUM = IDUM + 1
0036        IDUM = IDUM - 1
0037        GOTO 300
0038         STOP
0039         END
```

```
0001 CRL      SUBROUTINE TO DETERMINE RELATIVE HUMIDITY
0002 C
0003       SUBROUTINE RL(RECRH,CALTP,I,IDF,IH)
0004 ************************************************************
0005 *
0006       INCLUDE 'PRDCOM.SA'
0007 *
0008 ************************************************************
0009       CENT=273.16
0010       RAD=57.2957795
0011       IF(IH.NE.0)GO TO 4
0012       CALTK=CALTP+CENT
0013       A3=1.5*(RECRH-50.0)/RAD
0014       DUMMY=ZSIN(A3)/ZCOS(A3)
0015       DUMMY1=0.0355*CALTK-15.3315
0016       DUMMY2=DUMMY+33.75-0.087*CALTK+DUMMY1*ALOG10(CALRH)
0017       FK=DUMMY2/(0.08*ALOG10(CALRH)+0.27-0.002*CALTK)
0018       IH=1
0019       RETURN
0020 C     CHECK VALIDITY OF RH ORDINATE
0021     4 IF (OF(I) .EQ. 0.) GO TO 8
0022       IF(T(I) .LT. 233.16)GO TO 8
0023    43 IF(MOD(ICREN,2) .GT. 0)GO TO 46
0024 C     RH ELEMENT 476 (CARBON)
0025       DF=46.0-OF(I)
0026       IF(DF .LE. 0.0)GO TO 35
0027       CEE=64.8-CALRH/2.51
0028       R=100.0-CALRH
0029       AA=1.66+DF/33.0
0030       GO TO 36
0031    35 CEE=CALRH/1.45-5.198
0032       R=10.0-CALRH
0033       AA=0.5-DF/10.0
0034    36 DUMMY=DF/CEE
0035       DUMMY1=ABS(DUMMY)
0036       DUMMY2=POWER(DUMMY1,AA)
0037       F40M=CALRH+R*DUMMY2
0038       DUMMY=T(I)-233.15
0039       F(I)=F40M+SQRT(DUMMY)*OF(I)*(F40M-33.0)/1050.0
0040       GO TO 6
0041 C     RH ELEMENT 418 (LITHIUM)
0042  46   A3=0.02617994*(OF(I)-50.0)
0043       DUMMY=((ZSIN(A3)/ZCOS(A3)+33.75-0.087*T(I)-(0.27-0.002*T(I))*FK
0044 &          )/(15.3315+0.08*FK-0.0355*T(I)))
0045       F(I)=POWER(10.0,DUMMY)
0046     6 IF(F(I) .LT. 0.0)GO TO 8
0047       IF(F(I) .LT. 99.5) GO TO 55
0048       DEWPT(I)=T(I)
0049       IF(F(I) .LT. 100.0) GO TO 55
0050       F(I)=100.0
0051    55 IDF=0
0052       DUMMY=(7.5*T(I)-2048.7)/(T(I)-35.9)
0053       DUMMY1=POWER(10.0,DUMMY)
0054       E=F(I)*0.0611*DUMMY1
0055       IF(F(I) .EQ. 100.0)GO TO 7
0056        DEWPT(I)=(237.3*ALOG10(E)-186.527)/(8.286-ALOG10(E))
0057        DEWPT(I)=273.15+DEWPT(I)
0058    7   TV(I)=T(I)/(1.0-0.379*E/PR(I))
```

```
0059          GO TO 9
0060        8 TV(I)=T(I)
0061          DEWPT(I)=0.0
0062          F(I)=0.0
0063          IDF=1
0064 9        RETURN
0065          END
```

```
0001 CTEMPCE  SUBROUTINE TO CALCULATE TEMPERATURES
0002       SUBROUTINE TEMPCE(CALTP,IERRF,I,ITEMP)
0003 *
0004 C     SUBROUTINE TO COMPUTE RADIOSONDE TEMPERATURES 405,419
0005 *
0006 *****************************************************************
0007 *
0008       INCLUDE 'PRDCOM.SA'
0009 *
0010 *****************************************************************
0011       IERRF=0
0012       CENT=273.16
0013       IF(DT(I) .EQ. 0 .AND. ITEMP .NE. 0) GO TO 10
0014       IF(ITEMP.NE.0)GO TO 53
0015       DUMMY=2.0*RECTP
0016       DUMMY1=POWER(DUMMY,.996626)
0017       R30=-48000.0+8960200.0/DUMMY1
0018       CALTK=CALTP+CENT
0019       TLAST=CENT
0020       DUMMY=1510.0/CALTK
0021       DUMMY1=POWER(CALTK,2.95)
0022       AINV=RECTP*EXP(DUMMY)/(DUMMY1*(95.0-RECTP))
0023       ITEMP=1
0024       RETURN
0025    53 IF(ICRBN .GT. 1)GO TO 60
0026 C     TEMPERATURE ELEMENT 419
0027       FACTOR=AINV*(95.0/DT(I)-1.0)
0028    30 DUMMY=1510.0/TLAST
0029       EXPT=EXP(DUMMY)
0030       DUMMY=POWER(TLAST,2.95)
0031       DUMMY1=POWER(TLAST,1.95)
0032       T(I)=TLAST+(EXPT-FACTOR*DUMMY)/(1510.0*EXPT/(TLAST*TLAST)
0033 &       +2.95*FACTOR*DUMMY1)
0034       DUMMY=TLAST-T(I)
0035       IF(ABS(DUMMY) .LE. 0.02)GO TO 54
0036       TLAST=T(I)
0037       GO TO 30
0038    54 T(I)=T(I)-CENT
0039       T(I)=CENT+T(I)-T(I)*(1.0+T(I)*T(I)*T(I)*T(I)/9150625.0+(17.0+CALTP
0040 &      )*(17.0+CALTP)/16.0)/500.0
0041       GO TO 4
0042 C     TEMPERATURE ELEMENT 405
0043    60 DUMMY=2.0*DT(I)
0044       DUMMY1=POWER(DUMMY,.996626)
0045       REST=-48000.0+(8960200.0/DUMMY1)
0046       DUMMY=REST/R30
0047       DUMMY2=ALOG(DUMMY)
0048       DUMMY3=ALOG(10.)
0049       FACTOR=DUMMY2/DUMMY3
0050       T(I)=1.0/(3.298588E-3+1.07701E-3*FACTOR+5.89986E-5*(FACTOR**2))
0051 4     RETURN
0052 C     NO TEMPERATURE INPUT
0053    10 T(I)=-999.99
0054       RETURN
0055       END
```

```
0001 CWINDS   SUBROUTINE TO CALCULATE WINDS
0002 C
0003       SUBROUTINE WINDS(MAX,HGMDT,DISSFC,MIN)
0004 ************************************************************************
0005 C
0006       INCLUDE 'PRDCOM.SA'
0007 C
0008       DIMENSION V(125),U(125)
0009       RAD=57.29578
0010       R=6371.
0011       SPD(1)=VSFC
0012       DIR(1)=DSFC
0013       ANG=DSFC/RAD
0014       V(1)=ZCOS(ANG)*SPD(1)
0015       U(1)=ZSIN(ANG)*SPD(1)
0016       C=16.66667
0017       DISP=DISSFC/1000.
0018       DUMMY=AZ(1)/RAD
0019       XX(1)=DISP*ZSIN(DUMMY)
0020       YY(1)=DISP*ZCOS(DUMMY)
0021       ISTART=2
0022       IF(MIN .NE. 1)ISTART=MIN
0023       DO 10 M=ISTART,MAX
0024       HEIGH=HGP(M)-HGMDT
0025       Z=0.001*ABS(HEIGH)
0026       AZR=AZ(M)/RAD
0027       IF(EL(M) .EQ. 0.0) EL(M)=0.0001
0028       DUMM1=EL(M)
0029       DUMMY=ABS(DUMM1)
0030       ELR=DUMMY/RAD
0031       DUMMY=(1.0+Z/R)/ZCOS(ELR)
0032       DUMMY1=DUMMY**2-1.0
0033       DUMMY2=SQRT(DUMMY1)
0034       DUMMY3=ATAN(DUMMY2)
0035       DISP=R*(DUMMY3-ELR)
0036       XX(M)=DISP*ZSIN(AZR)
0037       YY(M)=DISP*ZCOS(AZR)
0038 10    CONTINUE
0039       MM2=MAX-2
0040       MM1=MAX-1
0041       MP1=MIN+1
0042       MP2=MIN+2
0043       U(MP1)=C*(XX(MP2)-XX(MIN))/(TIM(MP2)-TIM(MIN))
0044       V(MP1)=C*(YY(MP2)-YY(MIN))/(TIM(MP2)-TIM(MIN))
0045       U(MM1)=C*(XX(MAX)-XX(MM2))/(TIM(MAX)-TIM(MM2))
0046       V(MM1)=C*(YY(MAX)-YY(MM2))/(TIM(MAX)-TIM(MM2))
0047       DO 20 M=MP2,MM2
0048       IF(EL(M) .GT. 10.0)GO TO 21
0049       INP=M+2
0050       INM=M-2
0051       GO TO 22
0052    21 INP=M+1
0053       INM=M-1
0054    22 DUMMY=TIM(INP)-TIM(INM)
0055       U(M)=C*(XX(INP)-XX(INM))/DUMMY
0056    20 V(M)=C*(YY(INP)-YY(INM))/DUMMY
0057       DO 30 M=MP1,MM1
0058       DUMMY=U(M)*U(M)+V(M)*V(M)
```

```
0059        SPD(M)=SQRT(DUMMY)
0060        DUMMY=U(M)
0061        DUMMY1=V(M)
0062     30 DIR(M)=ADIR(DUMMY,DUMMY1)
0063        SPD(MAX)=SPD(MM1)
0064        DIR(MAX)=DIR(MM1)
0065        RETURN
0066        END
```

```
0001 C
0002       FUNCTION ZCOS(A)
0003 C
0004 C    THE 6809 CANT TAKE COS OF A NEG #
0005 C
0006       DUMMY=ABS(A)
0007       ZCOS=COS(DUMMY)
0008       RETURN
```

```
0001 C
0002       FUNCTION ZSIN(A)
0003 C
0004 C  THE 6809 WILL NOT TAKE THE SIN OF A NEG #
0005 C  THIS FUNCTION TAKES CARE OF THE PROBLEM
0006 C
0007       IF(A .GE. 0.0) GOTO 10
0008       A = -A
0009       ZSIN=SIN(A)
0010       ZSIN =-ZSIN
0011       RETURN
0012 10    ZSIN = SIN(A)
0013       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.

..... UNET-1 .....

CONDENSER/DECOMMUTATOR PROGRAM

UNIV. OF UTAH AUG 1982

INPUT DATA

STATION ID = 72
STATION HEIGHT = 4.0
ZERO AZIMUTH = 1
DATE = 3-10-82
SURFACE TEMP = 4.0
SURFACE RH = 66.0
SURFACE PRESS =1033.0
SURFACE WIND SPEED = 4.0
SURFACE WIND DIRECTION = 65.0
SONDE ID = 9897
REFERENCE ORD = 92.0
TEMP CAL ORD = 73.9
RH CAL ORD = 58.3
SURFACE TEMP ORD = 61.6
SURFACE RH ORD = 44.0
SONDE TYPE = 2
BURST CONTACT = 180.0

BAROSWITCH PRESSURE CALIBRATION TABLE

1:	1057.8	1046.2	1034.8	1023.6	1012.6	1001.2	989.8	978.8
9:	967.8	957.0	946.0	935.8	924.4	913.6	903.0	892.4
17:	881.6	871.2	860.8	850.2	839.6	829.4	819.0	808.6
25:	798.4	788.2	778.2	768.2	757.8	748.2	738.2	728.2
33:	718.4	708.8	699.2	689.6	680.0	670.8	661.0	651.6
41:	642.2	633.2	623.8	614.4	605.4	596.4	587.6	578.4
49:	569.8	561.0	552.4	543.8	535.4	527.0	518.4	509.8
57:	501.8	493.0	485.2	477.2	469.6	461.2	453.4	445.6
65:	437.8	430.0	422.6	415.2	407.8	400.4	392.8	385.8
73:	378.6	371.4	364.2	357.4	350.6	343.8	337.4	330.6
81:	324.0	317.4	310.8	304.8	298.6	292.4	286.2	280.2
89:	274.2	268.4	262.8	256.8	251.2	245.6	240.2	234.8
97:	229.4	224.2	219.0	214.0	208.8	203.8	199.2	194.2
105:	189.6	185.0	180.4	175.8	171.4	167.0	162.8	158.6
113:	154.4	150.2	146.0	142.4	138.2	134.6	130.6	127.0
121:	123.4	120.0	116.6	112.8	109.8	106.2	103.0	100.0
129:	96.8	93.6	90.4	87.6	84.6	81.8	79.0	76.2
137:	73.2	70.6	67.8	65.2	62.4	59.8	57.4	55.0
145:	52.4	49.8	47.2	44.8	42.4	40.0	37.4	35.0
153:	32.6	30.2	27.8	25.2	22.6	20.0	17.6	14.8
161:	12.2	9.2	6.6	3.8	0.0	0.0	0.0	0.0
169:	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
177:	0.0	0.0	0.0	0.0				

EFFECTIVE CONTACT NUMBER AT LAUNCH = 3.07
HIGHEST CONTACT NUMBER CALIBRATED = 164

BALLOON RELEASED AT 14:27:20.0

********** OUTPUT PROCESSING FOLLOWS **********

TEMP ORD FAILS DIFFERENCE TEST ** CHECK TEMP ORD AT 1.0000
TEMP ORD FAILS DIFFERENCE TEST ** CHECK TEMP ORD AT 2.0000

STATION 72 LAUNCH DATE 3-18-82 LAUNCH TIME14:27:20.8 GMT SONDE ID 9897

SURFACE CONDITIONS
PRESS 1033.8 MB BASE CAL = 30.8 C AT 73.9 GRD
TEMP 277.1 K HUMIDITY = 58.2 % AT 46.8 GRD
HUMY 66.8 %

TIME MIN	ALT GP MT	PRESS MB	LOG PRESS	TEMP DEG K	PTEMP DEG K	VTEMP DEG K	HUMTY PRCNT	DEWPT DEG K	SPECIF HUMTY	SPD MPS	DIR DEG	NS MPS	EW MPS
0.0	5.	1033.7	3.0144	277.1	274.5	277.60	65.6	271.2	0.0032	4.8	65.8	-1.6	-3.5
0.9	273.	1000.0	3.0000	275.2	275.2	275.68	66.7	269.6	0.0029	999.9	999.9	999.9	999.9
1.9	294.	997.4	2.9989	275.0	275.2	275.53	66.8	269.5	0.0029	999.9	999.9	999.9	999.9
2.0	583.	962.3	2.9833	276.1	279.1	276.15	0.0	0.0	0.0000	999.9	999.9	999.9	999.9
3.0	869.	928.9	2.9680	275.4	281.2	275.64	33.1	268.8	0.0016	999.9	999.9	999.9	999.9
3.9	1125.	900.0	2.9542	276.2	284.6	276.60	49.9	266.7	0.0027	999.9	999.9	999.9	999.9
4.0	1144.	897.8	2.9532	276.2	284.9	276.68	51.2	267.1	0.0027	999.9	999.9	999.9	999.9
5.0	1424.	867.5	2.9383	277.0	288.5	277.53	54.3	268.6	0.0031	9249.6	88.2	-1572.0	-9114.9
5.6	1588.	850.0	2.9294	274.8	287.9	275.31	53.7	266.5	0.0026	4071.4	89.2	-698.1	-4012.4
6.0	1717.	836.5	2.9225	273.2	287.5	273.57	53.2	264.8	0.0024	9.3	100.6	1.7	-9.1
7.0	2003.	807.1	2.9069	270.8	287.9	271.17	58.7	262.8	0.0020	9.5	103.4	2.2	-9.1
7.2	2073.	800.8	2.9031	270.9	288.8	271.19	0.0	0.0	0.0000	9.4	103.6	2.2	-9.0
8.0	2284.	779.8	2.8915	271.3	291.3	271.27	0.0	0.0	0.0000	9.8	103.9	2.2	-8.6
9.0	2574.	751.8	2.8756	270.2	293.3	270.58	38.5	258.1	0.0016	0.0	100.0	1.4	-7.8
10.0	2841.	726.8	2.8618	268.2	293.9	268.49	47.7	258.8	0.0017	7.4	88.6	-0.1	-7.3
11.0	3114.	701.2	2.8478	266.2	294.6	266.55	65.4	260.8	0.0021	9.2	72.1	-2.7	-8.7
11.0	3127.	700.0	2.8471	266.2	294.7	266.51	65.4	260.8	0.0021	9.4	71.5	-2.9	-8.8
12.0	3398.	676.1	2.8300	265.3	296.7	265.62	65.1	259.9	0.0020	13.8	62.5	-5.9	-11.4
13.0	3668.	652.9	2.8149	262.9	297.0	263.22	64.6	257.6	0.0017	15.4	56.8	-8.3	-12.7
14.0	3931.	631.0	2.8000	260.8	297.5	261.04	65.0	255.6	0.0015	15.6	62.3	-7.2	-13.7
15.0	4195.	609.5	2.7850	259.1	298.4	259.29	67.3	254.3	0.0014	13.7	62.8	-6.3	-12.0
15.5	4313.	600.0	2.7782	258.4	299.0	258.63	68.7	254.0	0.0014	13.6	62.2	-6.2	-11.9
16.0	4444.	589.7	2.7707	257.7	299.7	257.91	70.3	253.5	0.0014	13.5	62.5	-6.1	-11.9
17.0	4696.	570.3	2.7561	256.3	300.9	256.49	75.7	253.0	0.0013	14.7	72.5	-4.3	-13.9
18.0	4960.	550.6	2.7408	254.6	301.9	254.78	78.5	251.6	0.0011	14.8	72.9	-4.2	-14.8
19.0	5220.	531.7	2.7257	253.1	303.2	253.18	0.0	0.0	0.0000	17.5	71.9	-5.3	-16.5
20.0	5487.	512.8	2.7100	251.6	304.5	251.68	0.0	0.0	0.0000	21.1	69.5	-7.3	-19.7
20.7	5673.	500.0	2.6990	250.6	305.5	250.66	1.0	0.0	0.0000	22.3	65.6	-9.1	-20.2
21.0	5763.	493.9	2.6937	250.1	305.9	250.21	35.1	243.6	0.0007	23.0	63.9	-10.0	-20.5
22.0	6043.	475.4	2.6778	248.2	307.0	248.32	53.6	241.6	0.0006	24.4	58.9	-12.5	-20.8
23.0	6323.	457.3	2.6602	245.8	307.4	245.89	62.9	240.9	0.0005	23.8	58.6	-12.3	-20.2
24.0	6689.	439.4	2.6429	243.4	307.9	243.52	69.2	239.6	0.0005	22.2	59.3	-11.2	-19.0
25.0	6893.	422.1	2.6255	241.0	308.3	241.86	94.4	240.4	0.0006	23.4	63.1	-10.5	-20.7
26.0	7159.	406.5	2.6090	239.0	309.1	239.86	97.0	238.7	0.0005	22.9	62.6	-10.4	-20.2
26.4	7271.	400.0	2.6021	238.2	309.5	238.24	92.4	237.4	0.0004	23.2	63.6	-10.2	-20.7
27.0	7442.	390.3	2.5914	236.9	310.0	237.00	85.3	235.4	0.0004	23.8	65.0	-10.0	-21.5
28.0	7730.	374.4	2.5733	234.3	310.3	234.37	83.3	232.6	0.0003	25.4	71.1	-8.2	-24.0
29.0	8023.	358.6	2.5547	232.0	310.9	231.97	0.0	0.0	0.0000	24.1	69.5	-8.3	-22.4
29.6	8188.	350.0	2.5441	230.6	311.2	230.56	0.0	0.0	0.0000	25.2	68.6	-9.1	-23.3
30.0	8290.	344.8	2.5375	229.7	311.4	229.69	0.0	0.0	0.0000	25.8	68.8	-9.6	-23.8
31.0	8557.	331.3	2.5202	227.2	311.6	227.24	0.0	0.0	0.0000	28.1	66.5	-11.1	-25.6
32.0	8826.	318.1	2.5026	224.8	311.8	224.77	0.0	0.0	0.0000	33.2	66.0	-13.4	-30.2
33.0	9096.	305.2	2.4846	222.4	312.2	222.41	0.0	0.0	0.0000	30.5	66.6	-12.0	-27.9
33.4	9208.	300.8	2.4771	221.6	312.6	221.63	0.0	0.0	0.0000	28.3	64.6	-12.0	-25.4
34.0	9352.	293.4	2.4675	220.6	313.2	220.62	0.0	0.0	0.0000	25.6	61.4	-12.1	-22.4
35.0	9597.	282.5	2.4509	218.6	313.7	218.59	0.0	0.0	0.0000	29.5	59.2	-15.0	-25.2
36.0	9870.	270.6	2.4323	216.2	314.0	216.17	0.0	0.0	0.0000	32.7	65.9	-13.2	-29.7
37.0	10126.	259.8	2.4147	214.7	315.5	214.67	0.0	0.0	0.0000	34.1	65.6	-14.0	-31.0
38.0	10368.	250.0	2.3979	213.5	317.2	213.49	0.0	0.0	0.0000	40.0	61.5	-19.0	-35.0
38.8	10390.	249.5	2.3971	213.4	317.3	213.43	0.0	0.0	0.0000	40.3	61.3	-19.2	-35.2

39.0	10638.	239.4	2.3791	212.7	320.0	212.70	0.0	0.0	0.0000	42.0	63.2	-19.2	-30.1
40.0	10918.	228.9	2.3596	212.8	323.0	211.98	0.0	0.0	0.0000	38.0	65.1	-15.9	-34.4
41.0	11175.	219.6	2.3416	211.4	326.0	211.38	0.0	0.0	0.0000	39.0	64.6	-16.6	-35.1
42.0	11434.	210.6	2.3235	211.7	330.3	211.66	0.0	0.0	0.0000	43.0	69.6	-14.9	-40.2
43.0	11690.	202.1	2.3056	213.2	336.7	213.20	0.0	0.0	0.0000	48.9	74.5	-18.9	-39.3
43.2	11756.	200.0	2.3010	213.5	338.2	213.53	0.0	0.0	0.0000	39.4	74.3	-18.5	-37.8
44.0	11957.	193.7	2.2871	214.6	342.9	214.55	0.0	0.0	0.0000	34.6	73.8	-9.6	-33.2
45.0	12233.	185.3	2.2680	214.5	347.1	214.46	0.0	0.0	0.0000	36.6	73.9	-18.8	-35.0
46.0	12586.	177.5	2.2491	215.3	352.9	215.32	0.0	0.0	0.0000	29.1	76.8	-6.5	-28.3
47.0	12773.	170.1	2.2300	215.9	358.1	215.92	0.0	0.0	0.0000	24.8	79.7	-4.3	-24.3
48.0	13049.	162.9	2.2119	215.6	362.1	215.57	0.0	0.0	0.0000	34.8	74.9	-9.8	-33.5
49.0	13326.	155.9	2.1937	214.7	365.2	214.75	0.0	0.0	0.0000	35.3	71.1	-11.3	-33.3
49.8	13567.	150.0	2.1781	214.7	369.2	214.70	0.0	0.0	0.0000	47.0	87.6	-1.9	-46.8
50.0	13611.	149.0	2.1738	214.7	369.9	214.69	0.0	0.0	0.0000	49.4	89.7	-0.1	-49.3
51.0	13942.	142.6	2.1548	293.6	512.2	299.57	31.3	276.0	0.0334	26.7	80.2	-4.5	-26.2
52.0	14282.	136.7	2.1358	255.9	451.9	256.25	30.6	242.8	0.0022	17.8	39.4	-13.6	-11.2
53.0	14578.	131.1	2.1176	225.4	402.8	225.43	0.0	0.0	0.0000	26.5	81.4	-3.9	-26.1
54.0	14846.	125.8	2.0995	214.7	388.2	214.68	0.0	0.0	0.0000	35.5	88.3	-5.9	-34.9
54.1	14884.	125.0	2.0969	214.6	388.7	214.58	0.0	0.0	0.0000	35.9	79.9	-6.2	-35.2
55.0	15115.	120.5	2.0809	214.0	391.8	214.03	0.0	0.0	0.0000	30.3	77.5	-8.2	-37.3
56.0	15412.	114.9	2.0603	213.5	396.2	213.53	0.0	0.0	0.0000	21.8	84.3	-2.1	-21.6
57.0	15708.	109.6	2.0397	213.0	400.6	212.97	0.0	0.0	0.0000	28.4	84.4	-1.9	-28.2
58.0	16011.	104.4	2.0186	212.1	404.4	212.05	0.0	0.0	0.0000	31.2	78.7	-6.0	-30.5
59.0	16264.	100.2	2.0009	211.6	408.4	211.64	0.0	0.0	0.0000	28.4	84.3	-2.7	-28.1
59.0	16277.	100.0	2.0000	211.7	408.7	211.66	0.0	0.0	0.0000	28.0	84.4	-2.6	-27.7
60.0	16529.	96.0	1.9823	212.0	414.2	212.03	0.0	0.0	0.0000	19.6	86.7	-1.0	-19.5
61.0	16831.	91.5	1.9612	212.0	419.9	212.03	0.0	0.0	0.0000	35.8	78.8	-6.9	-35.1
62.0	17141.	87.0	1.9395	211.1	424.1	211.08	0.0	0.0	0.0000	21.9	82.1	-2.9	-21.6
63.0	17452.	82.7	1.9177	211.4	430.8	211.36	0.0	0.0	0.0000	8.3	108.6	2.7	-7.8
63.7	17660.	80.0	1.9031	211.2	434.6	211.21	0.0	0.0	0.0000	15.7	92.9	0.8	-15.6
64.0	17734.	79.0	1.8979	211.2	436.0	211.16	0.0	0.0	0.0000	18.5	98.5	0.1	-18.4
65.0	18027.	75.4	1.8773	211.6	442.9	211.62	0.0	0.0	0.0000	26.1	84.8	-2.2	-25.9
66.0	18334.	71.7	1.8557	210.8	447.6	210.84	0.0	0.0	0.0000	36.0	79.2	-6.6	-35.2
66.5	18486.	70.0	1.8451	210.8	450.7	210.83	0.0	0.0	0.0000	29.4	79.4	-5.3	-28.8
67.0	18643.	68.2	1.8341	210.8	454.0	210.83	0.0	0.0	0.0000	22.5	79.8	-3.9	-22.1
68.0	18968.	64.8	1.8113	211.0	463.0	211.01	0.0	0.0	0.0000	16.3	92.1	0.6	-16.2
69.0	19290.	61.5	1.7887	212.8	472.0	212.75	0.0	0.0	0.0000	17.4	97.6	2.3	-17.1
69.5	19442.	60.0	1.7782	213.3	476.5	213.26	0.0	0.0	0.0000	13.4	109.4	4.5	-12.6
70.0	19592.	58.6	1.7677	213.8	480.8	213.76	0.0	0.0	0.0000	18.5	128.8	6.6	-8.1
71.0	19881.	55.9	1.7478	215.2	490.4	215.16	0.0	0.0	0.0000	11.9	113.2	4.7	-10.9
72.0	20218.	53.0	1.7246	215.4	498.5	215.48	0.0	0.0	0.0000	26.4	78.1	-5.4	-25.8
73.0	20563.	50.2	1.7007	214.8	504.9	214.78	0.0	0.0	0.0000	31.5	72.2	-9.5	-29.9
73.1	20589.	50.0	1.6990	214.8	505.5	214.78	0.0	0.0	0.0000	38.1	72.3	-9.0	-28.6
74.0	20883.	47.7	1.6787	214.8	512.4	214.82	0.0	0.0	0.0000	14.1	75.5	-3.4	-13.5
75.0	21211.	45.3	1.6561	215.2	520.9	215.16	0.0	0.0	0.0000	7.4	144.8	6.1	-4.2
76.0	21536.	42.9	1.6323	215.7	530.3	215.67	0.0	0.0	0.0000	25.2	87.1	-1.7	-25.1
77.0	21903.	40.6	1.6085	216.5	540.7	216.48	0.0	0.0	0.0000	28.6	94.3	2.2	-28.4
77.3	21997.	40.0	1.6021	216.9	544.1	216.88	0.0	0.0	0.0000	24.1	98.3	3.8	-25.7
78.0	22251.	38.4	1.5847	218.0	553.0	217.97	0.0	0.0	0.0000	20.3	113.5	8.1	-18.5
79.0	22574.	36.6	1.5629	220.0	566.4	220.04	0.0	0.0	0.0000	22.4	85.0	-1.9	-22.2
79.9	22854.	35.0	1.5441	221.5	577.3	221.52	0.0	0.0	0.0000	26.5	87.7	-1.0	-26.4
80.0	22875.	34.9	1.5427	221.6	578.1	221.64	0.0	0.0	0.0000	26.9	87.8	-0.9	-26.7
81.0	23220.	33.1	1.5196	222.4	589.0	222.41	0.0	0.0	0.0000	15.3	98.3	2.2	-15.1
82.0	23562.	31.4	1.4969	223.0	599.6	223.04	0.0	0.0	0.0000	33.8	64.0	-14.7	-30.3
82.8	23860.	30.0	1.4771	224.3	610.8	224.26	0.0	0.0	0.0000	32.8	78.9	-10.4	-30.1
83.0	23915.	29.7	1.4734	224.5	612.8	224.48	0.0	0.0	0.0000	31.7	72.2	-9.6	-30.1
84.0	24288.	28.1	1.4489	225.4	625.2	225.36	0.0	0.0	0.0000	26.1	88.1	-4.4	-25.6
85.0	24702.	26.4	1.4216	225.9	638.8	225.86	0.0	0.0	0.0000	36.2	66.3	-14.5	-33.8
85.9	25063.	25.0	1.3979	225.8	648.0	225.84	0.0	0.0	0.0000	9.6	82.6	-1.1	-9.4
86.0	25121.	24.8	1.3942	225.8	649.5	225.84	0.0	0.0	0.0000	5.9	98.6	0.9	-5.7
87.0	25496.	23.4	1.3695	225.3	658.6	225.32	0.0	0.0	0.0000	28.8	78.9	-5.3	-27.4
88.0	25872.	22.1	1.3447	224.4	666.7	224.39	0.0	0.0	0.0000	36.2	76.5	-8.4	-35.1
89.0	26238.	20.9	1.3205	223.6	675.2	223.64	0.0	0.0	0.0000	45.6	78.6	-8.9	-44.6
89.8	26531.	20.0	1.3010	223.4	683.1	223.48	0.0	0.0	0.0000	32.3	82.1	-4.4	-31.9
90.0	26614.	19.8	1.2956	223.3	685.4	223.33	0.0	0.0	0.0000	28.6	83.6	-3.1	-28.4
91.0	26974.	18.7	1.2717	224.7	700.6	224.73	0.0	0.0	0.0000	7.8	108.5	2.5	-7.3

92.0 27358. 17.6 1.2465 226.8 718.8 226.78 0.0 0.0 0.0000 999.9 999.9 999.9 999.9

*** RECORDED INSTRUMENT HUMIDITIES ***
*** LESS THAN 20 PRCNT NOT LISTED ***

*** CHD IN LIMITING ANGLES ***
*** DURING PART OF OBSERVATION ***

*** LAYER BELOW 836.5 MB HAS SUPER ADIABATIC LAPSE RATE OF 13.8 DEG/KM

*** LEVEL BELOW 836.5 MB *** POTENTIAL TEMPERATURE = 287.4 DEG K *** NOT INCREASING

*** LAYER BELOW 136.7 MB HAS SUPER ADIABATIC LAPSE RATE OF 118.6 DEG/KM

*** LEVEL BELOW 136.7 MB *** POTENTIAL TEMPERATURE = 451.8 DEG K *** NOT INCREASING

*** LAYER BELOW 131.1 MB HAS SUPER ADIABATIC LAPSE RATE OF 183.1 DEG/KM

*** LEVEL BELOW 131.1 MB *** POTENTIAL TEMPERATURE = 462.8 DEG K *** NOT INCREASING

*** LAYER BELOW 125.7 MB HAS SUPER ADIABATIC LAPSE RATE OF 48.8 DEG/KM

*** 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

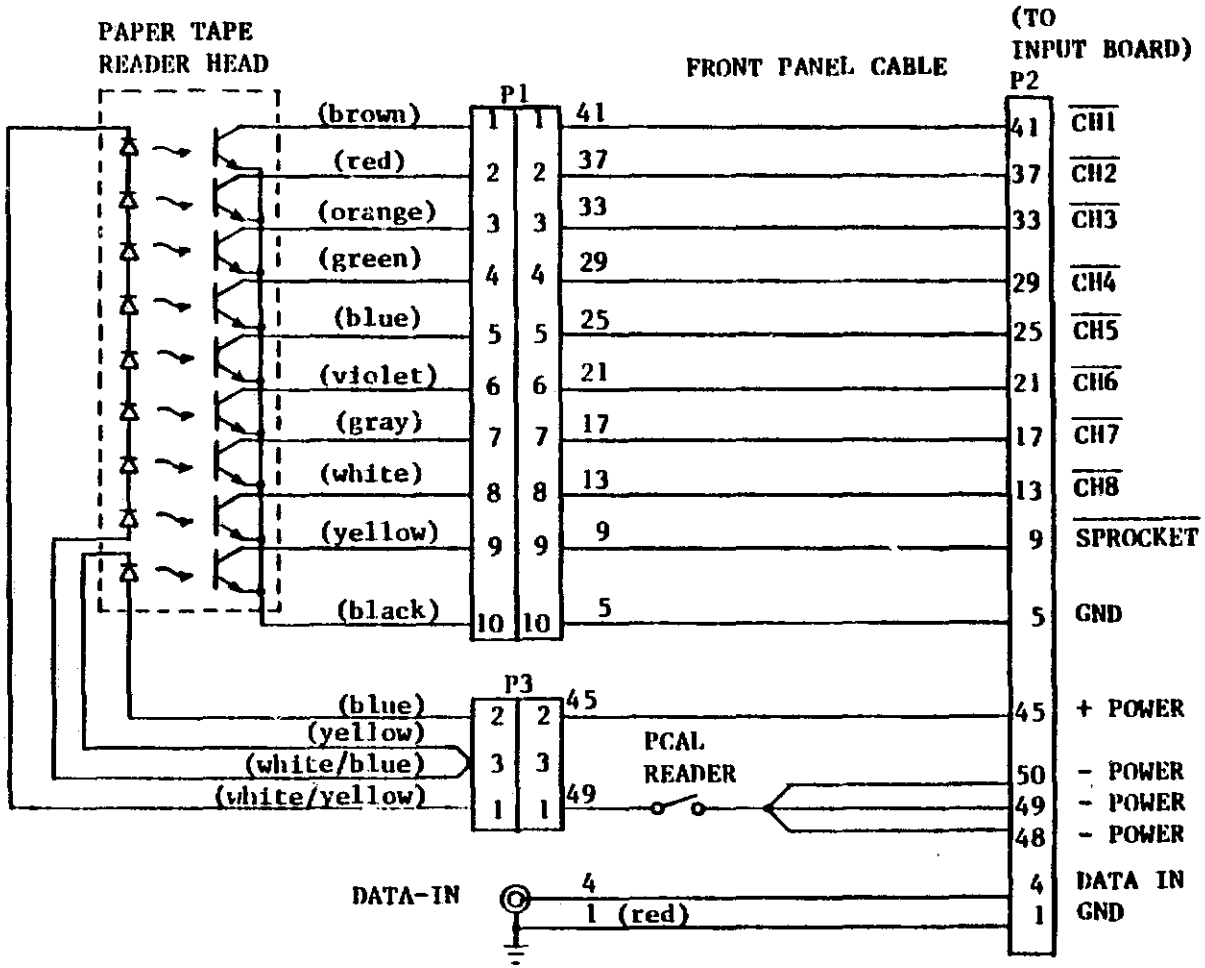


Fig. C-1. Paper tape reader and front panel cable C3.

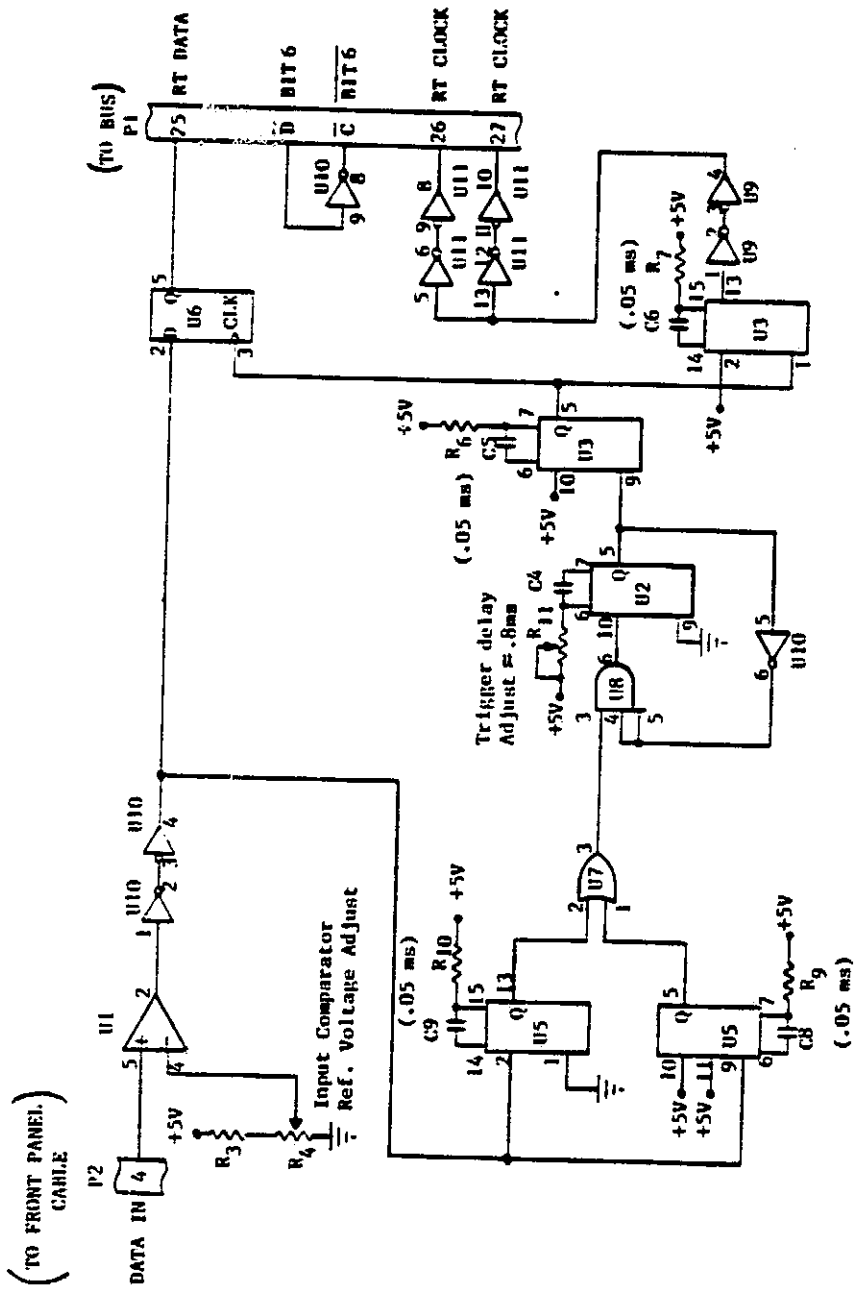


Fig. C-2. Biphase-to-level converter.

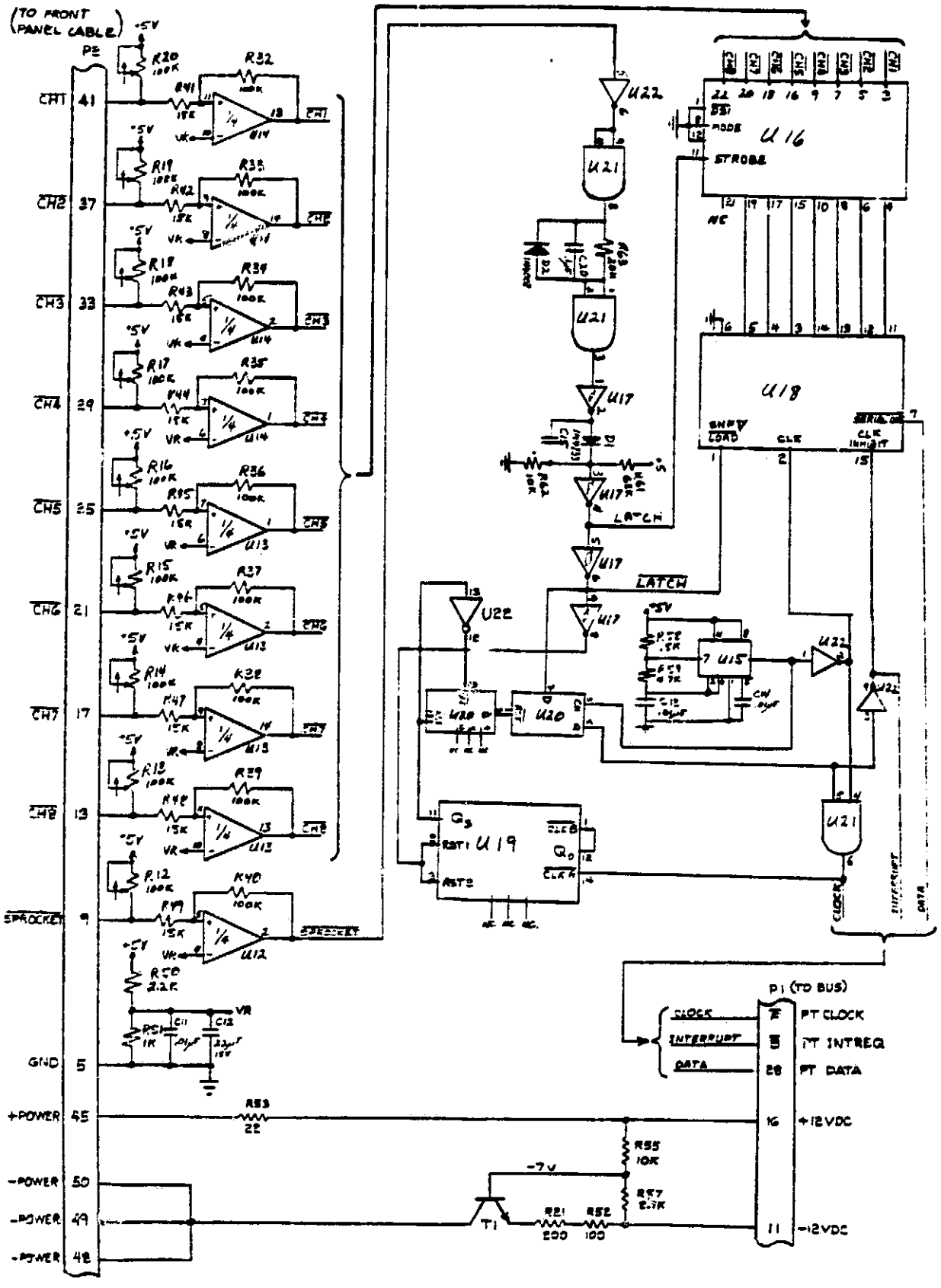
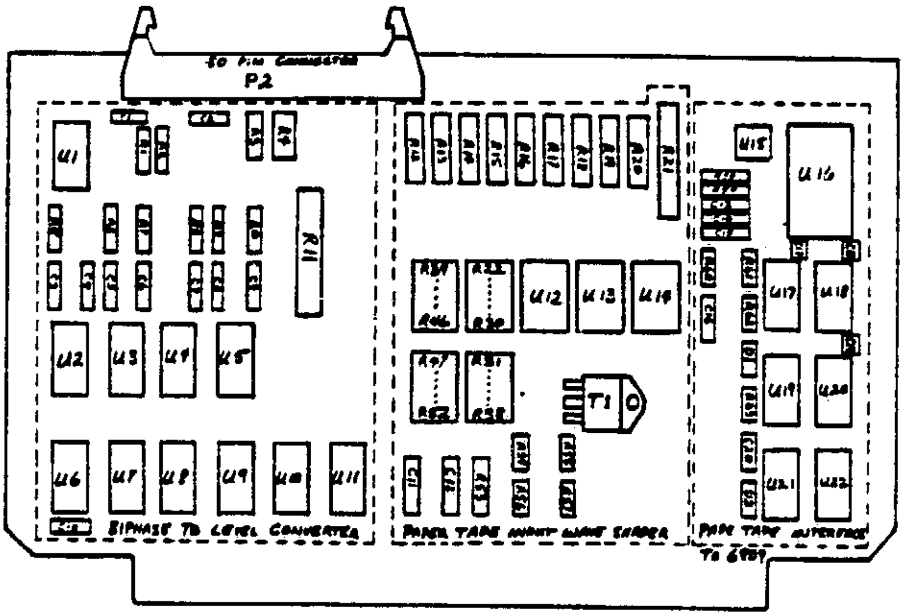


Fig. C-3. Input board — paper tape interface.

ORIGINAL PAGE IS
OF POOR QUALITY



—— ADJUSTMENTS ——

R4 INPUT COMPARATOR REF. VOLTAGE ADJUST
R11 DATA TRIGGER DELAY ADJUST
R12 SPROCKET INPUT LINE ADJUST
R18 CHANNEL 8 " " "
R14 " 7 " " "
R15 " 6 " " "
R16 " 5 " " "
R17 " 4 " " "
R18 " 3 " " "
R19 " 2 " " "
R20 " 1 " " "
R21 BIAS ADJUS FOR T1

—— SPECIFICATIONS ——

U1, 12, 13, 14 LM339N
U2, 3, 4, 5 74LS123N
U6, 20 74LS74N
U7 7432N
U8 74LS11N
U9, 10, 11 74LS04N
U17 74LS14N
U18 74LS165PC
U19 74LS90N
U21 74LS08N
U22 7404N
U15 555TC
U16 D8212
R1, 3 5.1K 5%
R2 220 5%
R4 5K POT 1TURN 3603502B
R5, 51 1K 5%

R6, 7, 8, 9, 10 1.5K 5%
R11 5K POT 20 TURN BOURNS 275-780-502
R12, 13, 14, 15, 16, 17, 18, 19, 20 100K POT 10 TURN WESTON
R21 200 POT 20 TURN FAIRCHILD 6023P-1-201
R22-R31, R62, 55 10K 5%
R32-R40, R60 100K 5%
R41-R49 15K 5%
R50 2.2K 5%
R52 100 5%
R53 22 5%
R54, 56 3K 5%
R57 27K 5%
R61 68K 5%
R58 510 5%
R59 4.7K 5%
R63 20K 5%
C1, 2, 10 .1µF
C3, 4 1.6µF 35V
C5, 6, 7, 8, 9 .1µF 50V
C11 .01µF 200Vdc
C12 22µF 15Vdc
C13 .01µF
C14 .01µF ±10%
C15 .1µF 50V
C16 100µF 35V
C17, 18, 19 .01µF
D1 1N4733
D2 1N4007
T1 3055 TEXAS INSTRUMENTS
P2 50 PIN CONNECTOR
RTI CARD MEX68WW

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-K 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-

---

* "METROC-K Algorithms," Forrest L. Staffanson, University of Utah Engineering Report UTEC MR 79-161, Progress Report under Contract NAS6-2627, to NASA Wallops Flight Center, November 1979 (revised January 1980). Also comparison reports UTEC MR 77-111 and UTEC MR 79-029.

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.

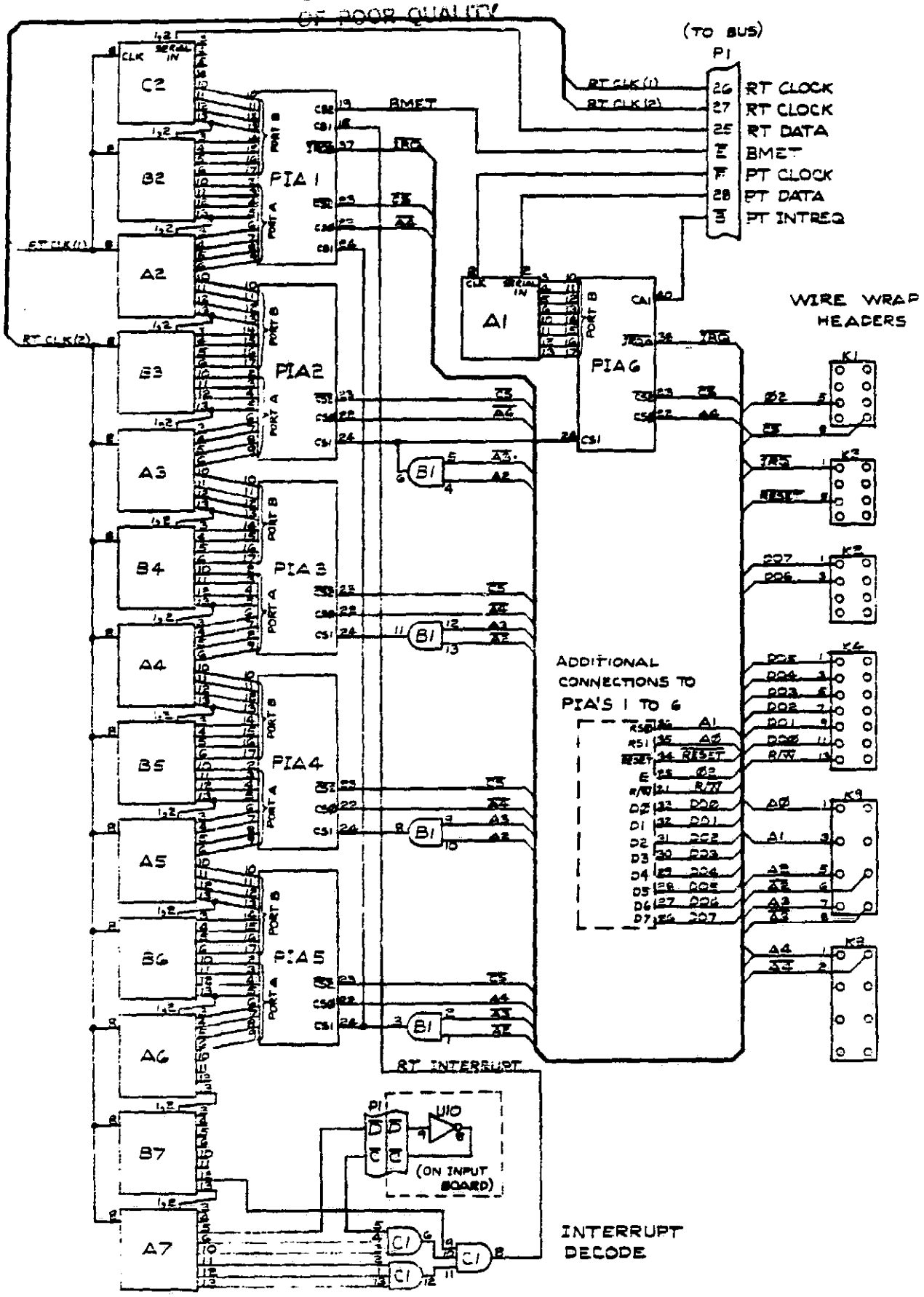
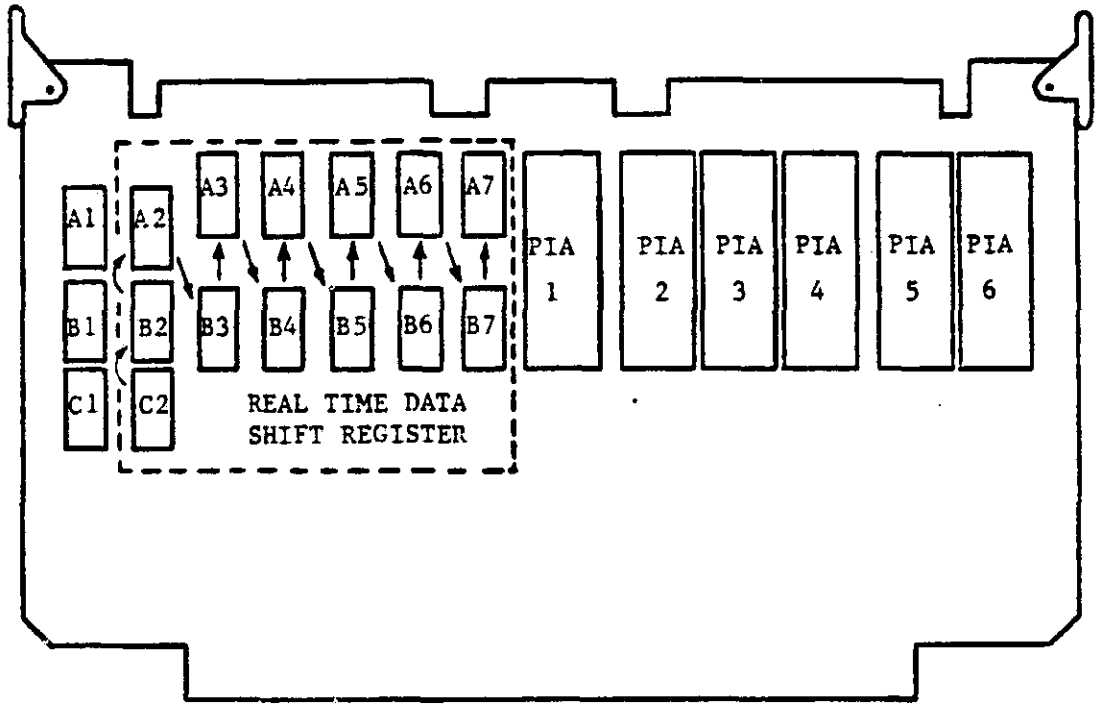
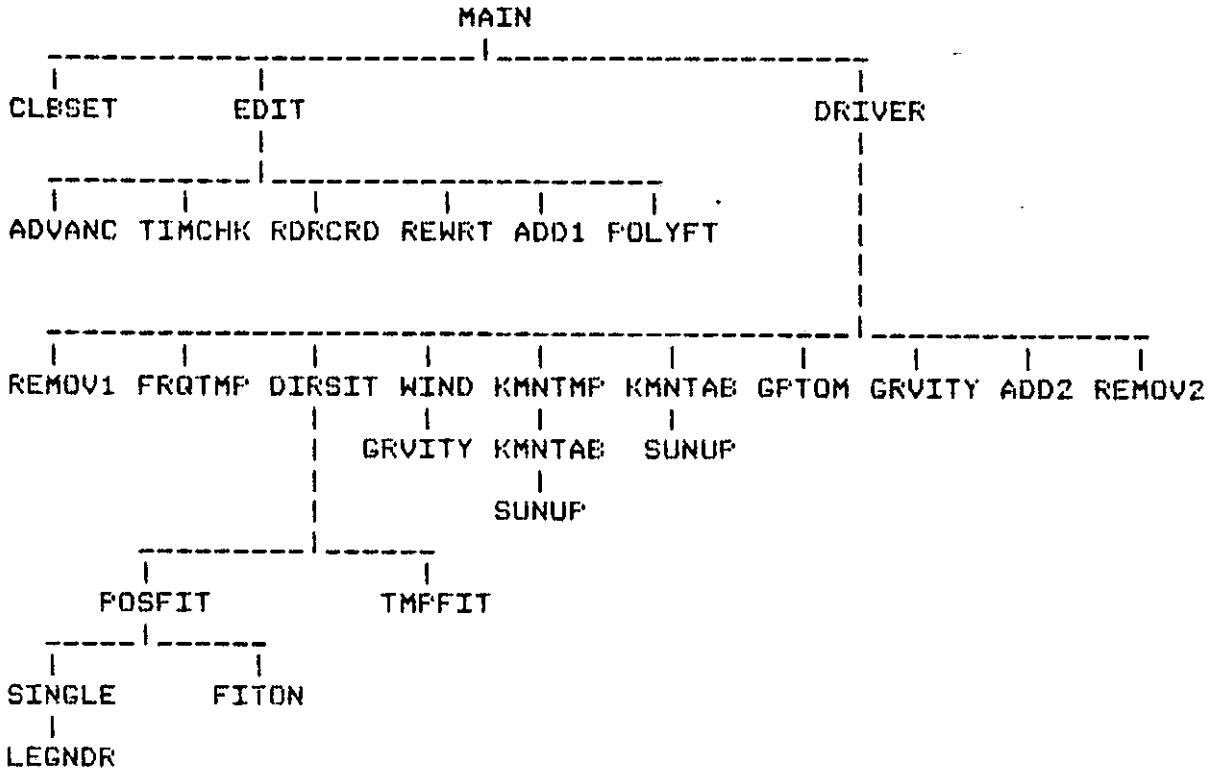


Fig. C-5. Interface board.



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

Fig. C-6. Interface board layout. Only the custom-installed components are shown; for a description of the rest of the board, see [10].



Subroutine Diagram of the METROC program

```
C-------------------------------------------------------------------
C   PURPOSE
C     This system of routines performs operations on
C     the double queue shown below. The two queues
C     "chase each other's tails".
C           ADD1 adds single data points to queue1.
C           ADD2 adds single data points to queue2.
C           REMOV1 removes single data points from queue1.
C           REMOV2 removes single data points from queue2.
C           REWRT replaces the frequency datum in an interval
C           of points at the head of queue1 with an "inert"
C           value.
C     Each 'data point' or 'place' in the queue contains
C     5 values. All data is added and removed as 5-vectors.
C
C                    --->          (empty)
C                  DDDDDDDDDD_____________EEEEEEEEEEE
C       queue 2    D            ^              ^          E  queue 1
C      (values     D         IHEAD2         ITAIL1        E (data points
C        from      D                                      E  from
C       DRIVER)    D                                      E  EDIT)
C                  DDDDDDD_______EEEEEEEEEEEEEEEEEEEEE
C                        ^       ^          <---
C                     ITAIL2    IHEAD1
C
C   DUMMY VARIABLES
C     D(5)           Data vector to be transferred between
C                    QFILE and calling routine
C     EMTY           Empty queue flag. Values are:
C                      1.=queue is empty; no point retrieved
C                     -1.=not empty
C     FULL           Full queue flag. Values are:
C                      1.=this queue is full; cannot add point
C                     -1.=not full
C
C   INTERNAL VARIABLES
C     IPASS          Flag to indicate the first call to ADD1
C                    so pointers can be initialized. Values:
C                      1= first pass; initialize pointers
C                      0= subsequent pass; don't initialize
C   COMMON VARIABLES
C     IHEAD1         Head pointer for queue1. Points to location
C                    of next point to be added to queue1.
C     IHEAD2         Head pointer for queue2. Points to location
C                    of next point to be added to queue2.
C     ITAIL1         Tail pointer for queue1. Points to location
C                    of next point to be removed from queue1.
C     ITAIL2         Tail pointer for queue2. Points to location
C                    of next point to be removed from queue2.
C     LNGTHQ         Length of queue= second dimension of QFILE
C     QFILE(5,1000)  Storage queue containing edited data points
C                    from EDIT and meteorological values from
C                    DRIVR1.
C                      QFILE(1,j)= time since launch(seconds)
C                      QFILE(2,j)= frequency datum (Hertz)
C                      QFILE(3,j)= X coordinate (meters)
C                      QFILE(4,j)= Y coordinate (meters)
C                      QFILE(5,j)= Z coordinate (meters)
```

```
C
C    CALLING ROUTINES - EDIT, DRIVER
C
C    COMMENTS
C      1.Subroutine ADD1 initializes the queue pointers
C        used by all the other queue access routines;
C        ADD2, REMOV1, REMOV2, REWRT. Therefore ADD1
C        must be called at least once before to be called.
C      2.LNGTHQ must be changed when QFILE is redimensioned
C      3.To set up queue for use in METROC:
C              IHEAD1=ITAIL1=(headstart)
C              IHEAD2=ITAIL2=1
C        where (headstart) is just large enough to prevent queue2
C        from catching up to queue1 in DRIVER. Finding an
C        adequate headstart may require repeated trials with
C        real flight data, so prgm may need modification to
C        allow the initial values to be changed at run time.
C
C    CHANGES
C      ADD1 is a new routine and was not part of METROC-K.
C
C    COMPILED 6/20/82
C    REFERENCES
C-----------------------------------------------------------------------
      SUBROUTINE ADD1 (D,FULL)
      INCLUDE 'RCOMMON.SA:1'
      DIMENSION D(5)
      DATA IPASS/ 1/
      IF (IPASS.EQ.1) THEN
        IHEAD1=10
        ITAIL1=10
        IHEAD2=1
        ITAIL2=1
        LNGTHQ=2000
        IPASS=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.ITAIL1) THEN
        FULL=1.
        RETURN
      END IF
      DO 10 I=1,5
        QFILE(I,IHEAD2)=D(I)
   10 CONTINUE
      IHEAD2=IHEAD2+1
      IF (IHEAD2.GT.LNGTHQ) IHEAD2=1
      END
```

```
0001         SUBROUTINE ADVANC (JJ,TSTOP,TLANCH,TGMDAQ,MASK)
0002 *
0003 ******************************** COMMON BLOCK ********************************
0004 *
0005              INCLUDE 'COMMON.SA'
0006 *
0007 ******************************************************************************
0008 C
0009 C            ADVANCE, 5 NEW RAW DATA POINTS.
0010 *
0011 C
0012           MASK=0
0013 C
0014         DO  2 JI=1,5
0015         J5= JI+5
0016          TIME(JI)=TIME(J5)
0017          FREQ(JI)=FREQ(J5)
0018          AZ(JI)=AZ(J5)
0019          EL(JI)=EL(J5)
0020       2 CONTINUE
0021 *
0022 ****     HANDLE REAL TIME DATA ***********************************************
0023 4887   DO 3 J=6,10
0024 3223   IF(IFLAG .GT. 0)GOTO 3224
0025         CALL STOPER
0026         IF(ISTPER .NE. 1)GOTO 3223 ;IF STOP IS HIT ISTPER=1
0027         MASK=4
0028         RETURN
0029 3224   INTMSK=1
0030         IF(IBOT .GE. 150)IBOT=0   ;IBOT POINTS TO OLDEST NOT YET PROCESSED POI
0031         IDUM=IBOT*10
0032         IBOT=IBOT+1               ;THE IDATA ARRAY IS ONLY 75 POINTS LONG
0033         IFLAG=IFLAG-1
0034         DUMMY=IDATA(IDUM+9)
0035         DUMMY1=IDATA(IDUM+10)
0036         FREQ(J)=(DUMMY*1000)+DUMMY1
0037         DUMMY=IDATA(IDUM+6)
0038         DUMMY1=IDATA(IDUM+5)
0039         EL(J)=DUMMY1+(DUMMY*.01)
0040         DUMMY=IDATA(IDUM+8)
0041         DUMMY1=IDATA(IDUM+7)
0042         AZ(J)=DUMMY1+(DUMMY*.01)
0043         DUMMY=IDATA(IDUM+4)
0044         DUMMY1=IDATA(IDUM+3)
0045         SECS=DUMMY1+(DUMMY*.1)
0046         DUMMY=IDATA(IDUM+2)
0047         AMINU=DUMMY*60.+SECS
0048         HOURS=(IDATA(IDUM+1))*3600.
0049         TIME(J)=HOURS+AMINU-TLANCH  ;TIME IS IN SECONDS FROM LAUNCH
0050         IF(INTMSK .EQ. 1)GOTO 444
0051         CALL GETBCK
0052 444    INTMSK=0
0053         IF(TIME(J) .LT. TIME(J-1))GOTO 5
0054         DUMMY=TIME(J-1)+60.
0055         IF(TIME(J) .GT. DUMMY)GOTO 5
0056 *
0057 ********** REAL TIME DATA NOW IN PROGRAM **************
0058 2204   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)*.000001)       ;DATA COMES IN IN USEC
0064 *
0065         IF(FREQ(J) .LT. 4.8 .OR. FREQ(J) .GT. 200.)GO TO 5
0066         GO TO 3
0067 5       J=J-1
0068  3       CONTINUE
0069          IF(TIME( 9) .GT. TSTOP) GOTO 100
0070 *
0071 *                  INCREMENT TABLE AND ENTER GMD ANGLES
0072 *
0073 51       IF(TIME(4) .LT. VL(1,LIST)) GO TO 53
0074          IF(TIME(4) .LT. TGMDAQ) GO TO 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-----------------------------------------------------------------------
C     9/28/82  CHRIS BURKS
C     PURPOSE:
C        'CLBSET' MODIFIES THE TCBEAD AND RCSOND
C        ARRAYS FOR LATER USE BY THE SUBROUTINE  'FRQTMP'.
C        IT TAKES THE LOGARITHM OF RCSOND AND CONVERTS
C        TCBEAD FROM CENTIGRADE TO KELVIN.
C
C     LOCAL VARIABLES:
C        I           LOOP INDEX
C        SAV         TEMPORARY STORAGE
C
C     COMMON VARIABLES:
C        NCBEAD      NUMBER OF THERMISTOR CALIBRATION POINTS
C        NCSOND      NUMBER OF SONDE CALIBRATION POINTS
C        RCSOND(25)  SONDE CALIBRATION DATA
C        TCBEAD(10)  THERMISTOR CALIBRATION DATA
C
C     CHANGES MADE:
C        1.PUT ALL COMMON VARIABLES INTO FILE 'RCOMMON.SA'
C          AS DESCRIBED IN 'MDOS FORTRAN REFERENCE MANUAL', P.1-6
C        2.ADDED 'SAV' VARIABLE BECAUSE FUNCTIONS
C          WILL NOT ACCEPT SUBSCRIPTED VARIABLE ARGUMENTS
C        3.ADDED THIS DOCUMENTATION
C
C     MODIFICATION IS FINISHED
C     WILL COMPILE
C     HAS NOT BEEN RUN
C-----------------------------------------------------------------------
      SUBROUTINE CLBSET
      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,ISWTCH)
C=======================================================================
C     Provides smoothed sonde positions and temperatures as well as
C     their first and second derivatives (velocities and accelerations).
C
C    DUMMY VARIABLES
C      IDIRSIT A switch to indicate when smoothed data is available
C              to DRIVER and to control initialization. Values:
C                1=initialize coefficients in smoothing equations.
C                2=add raw data point to time and position arrays
C                  (raw data) in POSFIT.
C                3=add raw data point to time, position, and
C                  temperature arrays.
C                4=add points to all above arrays and process all
C                  derivatives; return smoothed data. (all arrays
C                  are filled - enough points available to start
C                  processing)
C      NDRST   Switch to indicate whether temperature data will be
C              processed. Possible values:
C                3=wind-only data run: therefore only position data
C                  will be smoothed
C                4=both position and temperature data will be smoothed.
C      LTHF1   Smoothing interval length to obtain velocity in POSFIT.
C      LTHF2   Smoothing interval length to obtain acceleration POSFIT.
C      LTHT    Smoothing interval length for temperature and its
C              first and second derivative.
C      TIN     Time of raw data point.
C      XIN(4)  Raw X,Y,Z and temperature coordinates that are input to
C              subroutine on this call.  On return XIN equals the raw
C              data at time TOUT.
C      XOUT(4) Smoothed X,Y,Z and optionally temperature
C              calculated at time TOUT.
C      VOUT(4) First derivative of position and optionally temperature
C              calculated at time TOUT.
C      AOUT(4) Second derivative of position and optionally temperature
C              calculated at time TOUT.
C      ISWTCH  Switch  Values:
C                 0=IDRSIT switch will not be changed by DRIVER.
C                 1=IDRSIT Switch will be set equal to 3 by DRIVER
C                   so that proper logic path will be taken.
C
C    INTERNAL VARIABLES
C      ICNTF1  Index into raw data arrays X,Y,Z indicating where
C              the raw data is to be stored
C      ICNTF2  Index into the midpoint arrays (XMID, YMID, ZMID)
C              and velocity arrays (XVM, YVM, ZVM) where new
C              data is to be stored
C      LTHTSK  Total number of frequencies to store for
C              synchronization with POSFIT output
C      NSKIP   Number of raw data points which will be passed to
C              TMPFIT in order to assure that the smoothed data
C              are calculated for the same time point.
C
C    CALLING ROUTINE - DRIVER
C
C    SUBROUTINES CALLED - POSFIT, TMPFIT
C
```

```
C   CHANGES
C     Identical to METROC-K version
C
C   COMPILED 8/22/83
C=====================================================================
      DIMENSION XIN(4),XOUT(4),VOUT(4),AOUT(4)
      GO TO (10,100,200,200)  IDRSIT
C
C   INITIALIZE
C
   10 ICNTT=0
      ICNTP1=0
      ICNTP2=1
      CALL POSFIT(IDRSIT,TIN,XIN,TOUT,XOUT,VOUT,AOUT,ICNTP1,ICNTP2,
&        LTHP1,LTHP2,ISLIDE,ISWTCH)
      IF(NDRST.EQ.4)  CALL TMPFIT(IDRSIT,TIN,XIN(4),TOUT1,XOUT(4),
&        VOUT(4),AOUT(4),ICNTT,LTHT,ISLICE)
      IDRSIT=2
      IF(NDRST.EQ.4) GO TO 30
C
C   WHEN ICNTP1=LTHP1 THE RAW DATA ARRAYS ARE FULL, THEREFORE THE
C   MIDPT AND VELOCITY ARRAYS CAN BE CALCULATED,
C
   15 IF (ICNTP1.EQ.LTHP1) GO TO 18
      ICNTP1=ICNTP1+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 TMPFIT(IDRSIT,TIN,XIN(4),TOUT1,XOUT(4),VOUT(4),AOUT(4),ICNTT,
&        LTHT,ISLIDE)
   20 ISWTCH=1
      RETURN
C
C   PROGRAM ASSUMES LTHP1 + LTHP2 WILL BE GREATER THAN LTHT,NSKIT =
C   HOW MANY RAW DATA POINTS WILL NOT BE PASSED TO THE TMPFIT
C   SUBROUTINES IN ORDER TO ASSURE THAT THE SMOOTHED DATA ARE
C   CALCULATED FOR THE SAME THIME POINT.
C
   30 NSKIP=(LTHP1+LTHP2-LTHT-1)/2
C
C   LTHTSK = TOTAL NUMBER OF FREQUENCIES TO STORE FOR
C   SYNCHRONIZATION WITH POSFIT OUTPUT.
C
      LTHTSK=LTHT+NSKIP
  100 IF (NSKIP.EQ.0) GO TO 15
      IF (ICNTP1.EQ.LTHP1) GO TO 120
      ICNTP1=ICNTP1+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.ICNTT.EQ.LTHTSK.AND.ICNTP1.LT.LTHP1) GO TO 250
      IF(ICNTT.EQ.LTHTSK) GO TO 240
      ICNTT=ICNTT+1
  240 CALL TMPFIT(IDRSIT,TIN,XIN(4),TOUT1,XOUT(4),VOUT(4),AOUT(4),ICNTT,
&        LTHT,ISLIDE)
  250 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DRIVER
C=====================================================================
C   PURPOSE
C     This subroutine takes as input the queue of edited data
C     points created by EDIT. For each point, it calculates
C     the uncorrected sensor temperature. Then smoothes the
C     temperature data along with the 3 coordinates of the
C     sonde (X,Y,Z). Calculates the first two derivatives for
C     all four variables. Calculates the winds and relative air
C     speeds. Calculates the corrected air temperature. The air
C     temperature at 100 meter levels is saved for the second
C     part of the subroutine. To conserve memory, the air temp-
C     -erature is stored in the queue in place of the edited
C     data points.
C        In the second part of the subroutine, initial values
C     of atmospheric pressure are found by 'tying' the lower
C     end of the temperature array to radiosonde data. Then
C     the subroutine backsteps through the temperature array,
C     from low altitude to high, calculating atmospheric
C     pressure and density and printing out the meteorological
C     data.
C
C   INPUTS
C     BRES   = 30 WORD ARRAY, RADIOSONDE PRESSURES (MILLIBARS)
C     BTMP   = 30 WORD ARRAY, RADIOSONDE TEMPERATURES (DEG. CELSIUS)
C     DRTEMP= 100 WORD ARRAY, DIRECT INPUT SENSOR TEMPERATURE
C     DRTIM  = 100 WORD ARRAY, DIRECT INPUT SENSOR TIME
C     HEADER= 14 WORD LAUNCH OPERATION LABEL
C     IWIND  = INTEGER SWITCH, 1 = RADAR DATA ONLY
C     LALT   = 30 WORD ARRAY, RADIOSONDE ALTITUDE IN DECAMETERS
C     NOTEMP= SWITCH, 1 = NO DIRECT TEMPERATURES, -1 TEMP ON CARDS
C     NPBL   = NUMBER OF RADIOSONDE DATA
C     NTPT   = TOTAL NUMBER OF DIRECT TEMPERATURE POINTS
C     TAPOG = TIME IN SECONDS AFTER LAUNCH OF MAXIMUM SONDE ALTITUDE
C     TIME   = TIME IN SECONDS AFTER LAUNCH OF PRESENT DATA POINT
C     ZAPOG = MAXIMUM SONDE ALTITUDE ENCOUNTERED IN RADAR DATA
C     ZDD    = UPWARD ACCELERATION COMPONENT OF SENSOR
C     ZS     = SMOOTHED ALTITUDE OF SENSOR (M)
C
C   OUTPUTS
C     HEADER, LALT, BTMP = PRINTED VALUE SAME AS INPUT VALUE
C     ACC     = 800 WORD ARRAY, VERTICAL ACCELERACTION AT 100M LEVELS
C     AIRSPD = VENTILATION VELOCITY OF SONDE (M/S) AIRSPEED
C     ALTD    = ALTITUDE AT 100 METER INTERVALS, 800 WORD ARRAY
C     ALTG    = COMPUTED BASELINE TIE-ON ALTITUDE
C     BETA    = EXPONENT IN LINEAR TEMPERATURE MODEL
C     BTPINT = ROCKETSONDE TEMPERATURE AT ZLOW
C     CABSB   = TEMPERATURE CORRECTION FOR RADIATION ABSORPTION (DEG C)
C     CAERO   = TEMPERATURE CORRECTION FOR AERODYNAMIC HEATING
C     CDYN    = TEMPERATURE CORRECTION FOR DYNAMIC LAG DEGREE CELSIUS
C     CEMIS   = TEMPERATURE CORRECTION FOR RADIATION EMMISSION (DEG C)
C     CK1, CK2, CK3, CK4 = KRUMINS COEFFICIENTS FOR TEMP, CORRECTION
C     CTOTAL = TOTAL TEMPERATURE CORRECTION
C     DAYNGT = FLAG INDICATES DAY OR NIGHT FOR SENSOR (DAY=1.,NIGHT=-1.)
C     DENS    = AIR DENSITY COMPUTED AT PRESENT 100M LEVEL
C     DRCTN   = WIND DIRECTION IN DEGREE AZIMUTH
C     DTP     = TEMPERATURE DIFFERENCE BETWEEN PRESENT AND LOWEST TEMP
C     EWW     = 800 WORD ARRAY, EAST-WEST WIND AT 100M LEVEL (M/S)
```

```
C      FV      = 800 WORD ARRAY, FALL VELOCITY AT 100M LEVEL (M/S)
C      G       = GRAVITY COMPUTED BY FUNCTION GRVITY
C      GAMMA   = LAPSE TIME (DEG CELSIUS/METER)
C      IL      = INDEX OF TEMPERATURES
C      K       = INDEX OF RAWINSONDE DATA
C      LEV     = INDEX VALUE OF BASELINE TIE-ON LEVEL IN RADIOSONDE
C      MIN     = TIME IN WHOLE MINUTES AFTER LAUNCH AT PRESENT ALTITUDE
C      SNW     = 800 WORD ARRAY, NORTHWARD WIND COMPONENT (100M LEVELS)
C      PRES    = ATMOSPHERIC PRESSURE (MB) COMPUTED AT PRESENT 100M
C      RSZM    = 30 WORD ARRAY, RADIOSONDE GEOMETRIC ALTITUDE
C      ISEC     = SECONDS CORRESPONDING TO MIN
C      TAIR    = CORRECTED THERMOMETER AIR TEMPERATURE (DEG. CELSIUS)
C      TCOR    = 800 WORD ARRAY, TOTAL THERMOMETRIC CORRECTION 100M:S
C      TI      = CORRECTED MEASURED AIR TEMPERATURE (DEGREES KELVIN)
C      TMPT    = 800 WORD ARRAY, CORRECTED MEASURED AIR TEMPERATURE (CELSIUS)
C      TMPTRS  = 1150 WORD ARRAY, AIR TEMPERATURE (DEG. K) STORED IN .5
C                SECOND LIST FOR BACK-PROCESSING OF DENSITY AND
C                PRESSURE
C      WINDE   = SMOOTH-DIFFERENTIATED RADAR EASTWARD WIND COMPONENT
C      WINDN   = SMOOTH-DIFFERENTIATED RADAR NORTHWARD WIND COMPONENT
C      WINT    = WIND MAGNITUDE (M/S)
C      WTAU    = TIME CONSTANT OF WIND SENSING SYSTEM (RARACHUTE)
C      XD      = SMOOTHED EASTWARD VELOCITY COMPONENT
C      XOLD    = 800 WORD ARRAY, UNCORRECTED EASTWARD WIND 100M LEVELS
C      YD      = NORTHWARD VELOCITY COMPONENT OF SENSOR (M/S)
C      YOLD    = 800 WORD ARRAY, UNCORRECTED NORTHWARD WIND 100 M:S
C      ZD      = UPWARD VELOCITY COMPONENT OF SENSOR (M/S)
C      ZOLD    = 800 WORD ARRAY, UNCORRECTED UPWARD SENSOR VELOCITY M/S
C      ZSKM    = ALTITUDE (KM)
C
C    INTERNAL VARIABLES
C      BI      = OCTAL, BLANKS
C      DENSS   = 800 WORD ARRAY, COMPUTED AIR DENSITIES AT 100M LEVELS
C      LTHF1, LTHF2, LTH = SMOOTHING INTERVAL LENGTHS
C      MXTMP   = MAXIMUM NUMBER OF ALTITUDE POINTS TOBE STORED AND
C                BACK-PROCESSED
C      PRESS   = 800 WORD ARRAY, COMPUTED ATMOSPHERIC PRESSURE 100M:S
C      R       = GAS CONSTANT
C                'RECINP' IS THE INPUT RECORD FOR ONE TIME POINT
C      RTOD    = DEGREES PER RADIAN
C      SCOPY   = 13 WORD ARRAY, HOLDS PREVIOUS VALUE FOR DATA POINT
C                WHEN COMPUTING TEMPERATURE AND X,Y,Z 1ST AND 2ND
C                DERIVATIVES.
C      TEMP    = SMOOTHED SENSOR TEMPERATURE INTERPOLATED TO 100M LEVEL
C      WM      = AMEAN MOLECULAR WEIGHT OF AIR
C      XDD     = EASTWARD COMPONENT OF SENSOR ACCELERATION
C      XS      = SMOOTHED EAST COORDINATE OF SENSOR POSITION (M)
C      ZDECRE  = INTERVAL BETWEEN STORED VALUES OF PROCESSED DATA
C
C    CALLING ROUTINE - MAIN
C
C    SUBROUTINES CALLED
C      GETEMP, FRQTMP, DIRSIT, WIND, KMNTMP, KMNTAB, GPTOM, GRVITY
C
C    COMMENTS = REFERENCE ?METROC-K ALGORITHMS? FOR MATHEMATICAL
C               FORMULAE
C      NONSTANDARD RETURN TO DRIVER WHEN
C       MID-INDEX SEARCH VALUE EQUALS UPPER OR LOWER BOUND IN FRQTMP
```

```
C      OR KMNTAB; OR END-OF-FILE ENCOUNTERED IN FLRDD.
C
C    CHANGES
C      This is an extensively modified version of the METROC-K
C      subroutine. The modifications made are:
C      1.The edited data points come from the queue via subroutine
C        REMOV1 instead of from tape via FLRDD.  Subroutine FLRDD
C        has been eliminated.
C
C=======================================================================
      INCLUDE 'RCOMMON.SA:1'
      DIMENSION  RSZM(30),DPNT(5),X(4)
      DIMENSION  XOUT(4),VOUT(4),AOUT(4) ;smoothed data from DIRSIT
C-----------------------------------------------------------------------
C     All real constants stored in variables
C-----------------------------------------------------------------------
      DATA  PRS/0./, DNS/0./, BI/0./
      DATA  LTHP1/41/,LTHP2/17/,LTHT/25/,ISLIDE/1/
      DATA  ZDECRE /-100.0/
      DATA  ZSLAST/0.0/
      DATA  MXTMP /1150/
      DATA  PI /3.1415926/          ;'pi'
      DATA  RTOD / 57.295780/       ;radians to degrees
      DATA  WM/ 0.0289644/, R/ 8.31432/
      DATA  R999P9/999.9/, R1P/1./, R60P/60./, R1E3/1000./
      DATA  RZERO/0./, R9P99/9.99/, R9P999/9.999/, R99P99/99.99/
      DATA  R1E4/1.0E4/, R1E11/1.0E11/, R273P1/273.16/
      DATA  R10P/10./, R0P5/0.5/, R3E3/3.0E3/, R1E2/1.0E2/
      DATA  R1E5/1.0E5/, R7E401/70000.01/
      DATA  TRUE/1./,FALSE/-1./     ;'if' statment recognises
C                                   ; >0 as true and <0 as false
      AIRSP0 = RZERO
      IPD = 0
      IJ = 0
      IK = 0
      II = 0
      NDRST = 4                     ;no. of dirsit variables is
      IF (IWIND)  NDRST=3           ;3 for winds-only mode.
      I = ZAPOG/(-ZDECRE)
      Z100M = I*(-ZDECRE)           ;first 100m level below apogee
      ZHIGH = Z100M
C
C-----------------------------------------------------------------------
C                 READ DIRECT TEMPERATURES
C
C     The returned values are:
C        NOTEMP = 1  if no direct temp are given
C                 -1  if direct temp are given
C        NTPT   = the total number of direct temp points given
C        DRTEMP = array containing the direct temp values
C        DRTIM  = array containing the corresponding time values
C-----------------------------------------------------------------------
      CALL GETEMP1
      IDRSIT=1
      LINES=50
      INDEX=0
      ISWTCH=0
      IHEAD=1
```

```
      TAIL=1
      OLDATA=FALSE
      WRITE (IO,2400) HEADER
      WRITE (IO,2420) HEADER
C
C-----------------------------------------------------------------
C     START OF EDITED DATA LOOP: GET PT FROM QUEUE
C-----------------------------------------------------------------
   20 CONTINUE
      CALL REMOV1(P,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-----------------------------------------------------------------
C     CONVERT 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,NTPT          ;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
&       *(T-DRTIM(I))/(DRTIM(I-1)-DRTIM(I))  ;current time
        IWIND = FALSE               ;be sure wind-only flag not set
      END IF
C-----------------------------------------------------------------
C     SMOOTH 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,VOUT,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)             ;common variables used by WIND
      ZS      = XOUT(3)
      TEMP    = XOUT(4)
      XD      = VOUT(1)
      YD      = VOUT(2)
      ZD      = VOUT(3)
      TEMPD   = VOUT(4)
      XDD     = AOUT(1)
      YDD     = AOUT(2)
      ZDD     = AOUT(3)
      TEMPDD  = AOUT(4)
      MIN=TIME/R60P                 ;minutes since launch for print
```

```
      ISEC=TIME-MIN*60              ;seconds since launch for print
      ZSKM=ZS/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 (IO,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
C-----------------------------------------------------------------
C     CALCULATE WINDS AND TEMP AT POINT JUST BELOW 100M LEVEL
C-----------------------------------------------------------------
        CALL WIND (WTAU,WINDN1,WINDE1, ;calculate winds and airspeed
&       AIRSP1,WINDT,DRCTN)
        IF (.NOT.IWIND)               ;if not a 'winds-only' run
&         CALL KMNTMP (TAIR1,CAERO1,CDYN1, ;find air temperature
&         CEMIS1,CABSB1,AIRSP1)
C-----------------------------------------------------------------
C     CALCULATE WINDS AND TEMP AT POINT JUST ABOVE 100M LEVEL
C-----------------------------------------------------------------
        CALL SWAP (SCOPY)             ;swap old data into XS,YS,...TEMPDD
        CALL WIND (WTAU,WINDN0,WINDE0, ;find wind components & airspeed
&       AIRSP0,WINDT,DRCTN)
        IF (.NOT.IWIND)               ;if not a 'winds-only' run
&         CALL KMNTMP (TAIR0,CAERO0,CDYN0, ;find air temperature
&         CEMIS0,CABSB0,AIRSP0)
        CALL SWAP (SCOPY)             ;swap old data back out of XS,YS,...
C-----------------------------------------------------------------
C     INTERPOLATE DATA TO 100M LEVEL
C-----------------------------------------------------------------
   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 = R9P99
            CK2 = R9P999
            CK3 = R999P9
            CK4 = R99P99
          ELSE
            CK1 = CK1 * R1E4
            CK3 = CK3 * R1E11
          END IF
          IF (IWIND)  THEN
            WRITE (IO,2010) ZSKM,CK1,CK2,CK3,CK4
          ELSE
            TSEN = SCOPY(5) + RATIO*(TEMP-SCOPY(5))
            TCOR = TAIR - TSEN
            CAERO = CAERO0 + RATIO*(CAERO1-CAERO0)
```

```
            CDYN   = CDYN0   + RATIO*(CDYN1-CDYN0)
            CEMIS  = CEMIS0  + RATIO*(CEMIS1-CEMIS0)
            CABSB  = CABSB0  + RATIO*(CABSB1-CABSB0)
            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,CABSB,
&             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      ;
 2000       FORMAT ('*** 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-------------------------------------------------------------------
C     RADIOSONDE TIE-ON
C-------------------------------------------------------------------
  100 CONTINUE
  160 IK=INDX
      WRITE (IO,2432)
 2432 FORMAT(/10X,'***** NO TEMPERATURE CORRECTION A',
&            'PPLIED OUTSIDE THE KRUMINS TABL',
&            'E RANGE (20-70 KM) *****'////)
      INDX=INDX-1
      IL=INDX
      IF (IWIND)  GO TO 1166
      DO 161 I = 1,NPBL
      ISTORE = LALT(I) * R10P
      ISTORE = GPTOM (ISTORE, XLAT) + R0P5
      RSZM(I) = ISTORE
  161 CONTINUE
      ALTG=RZERO
C
C    *SCAN DATA FOR ERRORS, LINEAR INTERPOLATION OF TEMPERATURE GAPS
C+       LESS THAN 3KM
C . . CHECK 3-KM GAP BETWEEN R/S AND ROCKETSONDE ....
        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 (1A1)
      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
      IF(K.GE.1) GO TO 1140
 1141 KK=K
      K=K+1
      IF(KK.LE. 0) KK=K+1
C... CHECK 2.5 DEGREE DIFF, BETWEEN ROCKETSONDE TEMP, AT ZLOS AND
C    THE CORRESPONDING R/S TEMP, IN THE R/S BRACKET....
 1142 BTPINT=BTMP(KK)+(BTMP(K)-BTMP(KK))*(ZLOW-RSZM(KK))/(RSZM(K)-
&            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),F8.2,10X,F10.2)
 1143 LEV=K
      ALTG=RSZM(K)
      GO TO 1165
C...   PREFORM LINEAR INTERPOLATION ON TEMP, TOFILL THE TEMP GAP WHERE
C      THE ALT. GAP BETWEEN R/S AND ROCKETSONDE IS LESS THAN 3KM ....
 1144 WRITE(IO,2163)
 2163 FORMAT(/10X,'*** THE FOLLOWING INFORMATION ARE RELA',
&           'TED TO INTERPOLATION DUE TO N',
&           'O 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.EQ.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 DE',
&           'NSITY COMPUTATION DUE TO FAILURE ON 2.5 DEG',
&           'REE TEST OF TEMPERATURE ***'/10X,  80('*')////)
      I=K-1
      STORE = LALT(I) * R10P
```

```
      IF(K.GT.1) WRITE( IO,2162) I,RSZM(I),STORE,BTMP(I)
C
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) GOTO 1167
      IXS=INDX
      GO TO 206
 1167 CONTINUE
C
      GAMMA = RZERO
      G = GRVITY (ZLOW)
      BETA = RZERO
      TAIR = DFILE(1,IL)
      BTMPK=BTMP(K)+R273P1
      PRES=BPRES(LEV)*(TAIR/BTMPK)**
&       ((-WM*GRVITY(ZLOW)*(ZLOW-RSZM(LEV)))/(R*(TAIR-BTMPK)))
      IXS=INDX
      NKPT = 1
      I = IL
      GO TO 180
C
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 = -WM*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
C
C                  NORMAL TERMINATION
C
  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
      PRS=BI
      DNS=BI
  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
      INDX=IXS
  207 CONTINUE
      IF(I.GT. INDX) GO TO 174
      PRS=PRESS(I)
      DNS=DENSS(I)
      IF(IWIND) DFILE(1,I)=DFILE(1,I)+R273P1
      TI(I) = DFILE(1,I) - R273P1
      NKPT = NKPT + 1
      LN=LN+1
      IF(LN.LE.40) GO TO 210
      LN = 0
      WRITE(IO,2462)
 2462 FORMAT (1H1,?   ALT       EWW       SNW       FV     TMP(C)      PRS?,
&          BX,?DNS      ACC      XOLD      YOLD      ZOLD     TCOR    GAMMA C/M ?,
&          ?   BETA    GRAV?)
      ALTD = ALTI(ZHIGH,ZDECRE,I)
  210 WRITE(IO,2463) ALTD,  EWW(I),SNW(I),FV(I),DFILE(1,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)
C
C      CONTINUE LOOPING UNTIL ALL DATA IS PROCESSED.
C
C      WHEN IWIND = 1 , THIS IS A WINDS ONLY CASE,
C      THE PROGRAM MUST NOT BE ALLOWED TO BACKSTEP THRU THE ARRAY OF
C      TEMPERATURES.        STATEMENT 170
C
C      INSTEAD REDUCE THE COUNT BY 1 AND GO TO STATEMENT 207.
C
  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
  175 CONTINUE
  176 CONTINUE
      RETURN
C
```

```
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),
&           3X, F4.2, 2X, F5.3, 2X, F6.2, 2X, F5.2, 6X, A1,4X,2(2X,F5.1))
 2010 FORMAT (' ',F4.1,50X,F4.2,2X,F5.3,2X,F6.2,2X,F5.2)
      END
```

```
      SUBROUTINE EDIT
C=====================================================================
C   PURPOSE
C     This subroutine retrieves the .1 second raw data from the
C     input queue as it becomes available, averages sets of NAVG
C     raw points into clean edited data points, and adds them
C     to the edited data queue.
C
C   CONSTANTS
C     DINERT       "inert" value. A large constant, alternately
C                  positive and negative, which is substituted for
C                  the frequency datum of noisy or reference
C                  points. The filter DIRSIT will "coast"
C                  (quadratically extrapolate) through these
C                  unaffected by their amplitude.
C     INOISE       =1. Value of dat-classification flag indicating
C                  noise
C     IREF         =2. Value of dat-classification flag indicating
C                  reference signal
C     ISIG         =3. Value of data-classification flag indicating
C                  thermistor signal
C     IUHB         Bottom index of upper half of DSTOR
C     IUHT         Top index of upper half of DSTOR
C     TSTOP1       Stop time plus 12; METROC requires a 12 second
C                  buffer at the end of the data to be processed
C
C   INTERNAL VARIABLES
C     BLWR         Lower bound of tracking gate (Hertz)
C     BND          Upper bound of each overlapping 10Hz band
C                  used in searching for signal
C     BUPR         Upper bound of tracking gate (Hertz)
C     D            Sonde frequency used to compare against bounds
C                  of tracking gate (temporary storage)
C     DIFERT       Difference between the absolute values of
C                  a point's frequency value and the "inert"
C                  value. Used to detect if point has been
C                  given "inert" value.
C     DSTOR(5,10)  A 10 point sample of raw .1 second input data
C                    DSTOR(1,j) = GMT time of data pt (seconds)
C                    DSTOR(2,j) = frequency datum (Hertz)
C                    DSTOR(3,j) = azimuth (degrees)
C                    DSTOR(4,j) = elevation (degrees)
C                    DSTOR(5,j) = range (meters)
C     DSUBST       Mean value of raw data points which lie
C                  in the tracking gate. Used as a substitute
C                  for points which lie outside the gate, and
C                  to adjust SIGLEV.
C     DTRF         Time at midpoint of reference interval. Used to
C                  calculate sums for least squares fit (same as TRF)
C     EDATA(5)     A clean edited data point ready to be stored
C                  in the edited data queue
C                    EDATA(1) = time since launch (seconds)
C                    EDATA(2) = frequency datum (Hertz)
C                    EDATA(3) = X coordinate of sonde (meters)
C                    EDATA(4) = Y coordinate of sonde (meters)
C                    EDATA(5) = z coordinate of sonde (meters)
C     EOD          End-of-data flag returned by ADVANC. Values:
C                    1= flight is over, no more raw data coming
```

```
C                  -1= more data coming
C      FULL        Full queue flag returned by ADD1. Values:
C                    1= edited data queue is full; current pt not stored
C                   -1= queue not full
C      I           General DO loop index
C      IB          Index to bottom half of DSTOR
C      IBND        Index of most populous 10Hz band in searching for
C                  signal.
C      IFLAG       Data-classification flag. Indicates whether
C                  edited data point is noise (=1), reference (=2),
C                  or thermistor signal (=3). Used for printout.
C      INGATE(10)  In MET signal tracking, the flag indicating
C                  whether each .1 second data point in DSTOR(3,j)
C                  is in (=1) or not in (=0) the signal tracking
C                  gate.
C      INTRVL      Number of points in the queue to be replaced
C                  with "inert" value, starting with last point
C                  put in.
C      ISKREF      Used to compute initial points of signal to be
C                  skipped when replacing reference data with
C                  "inert" value.
C      ISS         Signal point counter. Number of edited data
C                  points accepted as thermistor signal.
C      IT          Index to top half of DSTOR
C      J           General DO loop index
C      JR          Reference point counter. Counts number of ref
C                  points since last signal interval. JR=0 during
C                  signal interval.
C      JRF         Reference interval counter. Counts number of
C                  reference intervals processed (at least 5
C                  consecutive reference points constitute a
C                  reference interval)
C      KB          Number of points in DSTOR with frequency
C                  less than upper bound BND of 10 Hz band.
C      KBL         Number of points in DSTOR with frequency
C                  less than BND - 5 Hz
C      KBLL        Number of points in DSTOR with frequency
C                  less than BND - 10 Hz (not initialized
C                  until BND = 25)
C      KBND(32)    In searching for MET signal, population in
C                  29 overlapping 10 Hz bands: KBND(3)= 20-30 Hz,
C                  KBND(4)= 25-35 Hz,...,KBND(31)= 160-170 Hz
C      LOS         In tracking signal, number of edited noise
C                  points since last signal or reference point.
C                  (not used)
C      NAFTR       Verify interval counter. Counts signal points
C                  after a reference interval, verifying termination
C                  of the reference interval.
C      NGATE       1 plus the number of raw data points in DSTOR(2,j)
C                  whose frequencies lie within the signal tracking
C                  gate (SIGLEV+-HGATE)
C      REF         Mean frequency of a reference interval (Hertz),
C                  excluding the first and last points. Used to
C                  compute sums for least squares fit to ref function.
C      RS          Frequency value of the last reference point in
C                  a reference interval
C      RSUM        Sum of frequency values in a reference interval,
C                  excluding the first point in the interval. Used
```

```
C                  for computing mean value over the interval.
C      RT1         Time after launch of the second point in a reference
C                  interval (seconds)
C      RT2         Time after launch of the next to last point in
C                  a reference interval (seconds)
C      RT3         Time after launch of the most recent reference
C                  point received. Saved for RT2 if next point
C                  is also reference, discarded if last point in
C                  reference interval.
C      SUM(5)      Temporary sums for averaging raw .1 second data
C                  in bottom half of DSTOR to get edited point (EDATA)
C                      SUM(1)    (not used)
C                      SUM(2)= sum of frequencies in DSTOR(2,j)
C                      SUM(3)= sum of azimuths    in DSTOR(3,j)
C                      SUM(4)= sum of elevations  in DSTOR(4,j)
C                      SUM(5)= sum of ranges      in DSTOR(5,j)
C      SUMGTE      Accumulator for computing mean of frequencies
C                  within the gate when tracking signal.
C      TDSTRT      Elapsed time since launch of raw data (seconds)
C      TEST        Average of frequencies in bottom half of DSTOR
C                  Used to tell if 170<=TEST<=200.
C      TLAPS       Elapsed time since TSTART of raw data (seconds)
C                  Used to detect starting point.
C      TRF         Time at midpoint of reference interval, calculated
C                  between second and next to last edited pt in intrvl
C
C    COMMON VARIABLES
C      B(3)        Accumulated parameters used by POLYFT to
C                  compute quadratic least squares fit to
C                  reference data.
C                  B(1)= sum of (ref interval frequency)
C                  B(2)= sum of (ref interval frequency *
C                        ref interval midpoint time)
C                  B(3)= sum of (ref interval frequency *
C                       (ref interval midpoint time)**2)
C      BT(5)       Accumulated parameters used by POLYFT to
C                  compute quadratic least squares fit to
C                  reference data.
C                  BT(1)= number of reference intervals
C                  BT(2)= sum of (ref interval midpoint time)
C                  BT(3)= sum of (ref interval midpoint time **2)
C                  BT(4)= sum of (ref interval midpoint time **3)
C                  BT(5)= sum of (ref interval midpoint time **4)
C      HGATE       Half-width of signal gate used for tracking
C                  thermistor signal (Hertz)
C      IO1         Output device number for informative messages.
C                  Possible values:
C                     99= dummy device. No output occurs when used
C                    101= terminal display
C                    102= printer
C      IWIND       Winds-only flag. Possible values:
C                       1= radar data only will be processed
C                      -1= temperature data will also be processed
C      NAVG        Number of raw data points averaged to obtain an
C                  edited data point (cannot exceed 1/2 the dimension
C                  of DSTOR) Was 5 in original program.
C      SIGLEV      Center frequency of signal gate used for tracking
C                  thermistor signal (Hertz). Starts at approximate
```

```
C                   value (155) and is adjusted to follow signal.
C      TAPOG        Time in seconds after launch of maximum sonde
C                   altitude encountered
C      TSTART       Time in seconds after launch to start processing
C                   data (selected to occur after payload ejection)
C      TSTOP        Time in seconds after launch to stop processing
C                   data
C      TZERO        GMT launch time (seconds)
C      ZAPOG        Maximum sonde altitude (apogee) encountered (meters)
C
C    SUBROUTINES CALLED
C      ADVANC, TIMCHK, REWRT, RDRCRD, ADD1, POLYFT
C
C    CALLED BY - MAIN
C
C    TERMINATION CONDITIONS
C      If number of reference intervals is less than 3/ prints
C      'Error in  Edit - Less than 3 reference values available'
C      and stops
C
C      If terminal 'S' key is pushed in ADVANC/ prints
C      'End of real time processing'  and returns
C
C      If edited data queue is full/ prints 'Queue is full'
C      'End of real time processing' and returns
C
C    CHANGES
C      This is an extensively modified version of the METROC-K
C      subroutine EDIT. The modifications made are:
C      1.The raw data is input from the raw data queue via
C        subroutine ADVANC, instead of from tape via
C        subroutine INTAPE as in the old routine.
C      2.All parts dealing with FSK time have been deleted.
C      3.The "replacement A, E, R data" sections have been deleted.
C      4.The number of raw data points in the average,
C        previously 5, has been made adjustable with the variable
C        NAVG. This allows run-time experimentation to determine
C        the proper data rate to avoid exhausting queue capacity
C        before the end of the flight.
C      5.Edited data points are stored in the edited data queue
C        by subroutine ADD1, instead of being stored on tape as
C        in old routine. Array DFILE, which was used to accumulate
C        data for writing to tape, has been eliminated.
C      6.New subroutine REWRT accesses the queue to replace
C        reference dwell frequencies with "inert" values. In the
C        old routine this was done in array DFILE.
C      7.The order of statements in the subroutine has been changed
C        around to "untangle" it and eliminate unnecessary GO TO
C        statements.
C
C    COMPILED  8/16/83
C    REFERENCES
C=================================================================
      INCLUDE 'RCOMMON.SA:1'
      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), KBND(32), SUM(5)
C-----------------------------------------------------------------
```

```
C                INITIALIZE PARAMETERS
C-------------------------------------------------------------
      DATA NAFTR0/ 5/
      DATA ISIG/ 1/, IREF/ 2/, INOISE/ 3/
      IUHB=NAVG+1                ;lowest index in upper half of DSTOR
      IUHT=2*NAVG                ;top index in upper half of DSTOR
      TSTOP1=TSTOP+12.           ;prgm 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
      KBLL=0
      KBL=0
      LOS=0                      ;noise point counter
      ZAPOG=0.                   ;apogee altitude
      DO 30 I=1,3                ;sums and coefficients
        C(J)=0.                  ;for least squares fit
        B(J)=0.
        BT(J)=0.
   30 CONTINUE
      BT(4)=0.
      BT(5)=0.
C
C-------------------------------------------------------------
C                READ TO START OF DATA
C                AND FILL UPPER HALF OF DSTOR ARRAY
C-------------------------------------------------------------
   40 CONTINUE
      DO 50 I=IUHB,IUHT
        CALL ADVANC (            ;get point from input queue:
&          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
        TDSTRT=TLAPS-TSTART      ;elapsed time since TSTART
        IF (TDSTRT.LT.0.25) GO TO 40 ;not past start time? start over
   50 CONTINUE
C
C                The raw data points should come at .1 second
C                intervals. TIMCHK checks the time values and
C                corrects any dropouts. If there are too many
C                dropouts the program is stopped.
C
      CALL TIMCHK (DSTOR,NAVG,4)
C
C-------------------------------------------------------------
C                BEGINNING OF REAL TIME LOOP
C-------------------------------------------------------------
C                Each iteration the .1 second raw data
```

```
C                points in the top half of DSTOR are shifted
C                to the bottom half for processing, and a
C                new set is put in the vacated top half by
C                ADVANC.
C                ADVANC gets a point from the raw data
C                queue; if there are none it waits for one to
C                come in. If the terminal's S key has been
C                pushed,it returns EOD = 1, indicating that
C                real time processing is finished.
C                The .1 second raw data points are processed
C                in sets with NAVG points per set (in previous
C                version NAVG was 5). Later each set
C                will be averaged into a single edited data point.
C
   60 CONTINUE                    ;start of main loop
      DO 80 IB=1,NAVG             ;get a set of raw data pts
        IT=IB+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-----------------------------------------------------------
C                TRACKING GATE PROCESSING
C-----------------------------------------------------------
  664 CONTINUE
      BUPR=SIGLEV+HGATE           ;tracking gate upper bound
      IF(BUPR.GT.170.) BUPR=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.BUPR .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
```

```
        BND=20.
        IBND=1
        DO 662 I=1,31              ;search 29 overlapping 10Hz bands
          KB=0                     ;  20-30,25-35,...,160-170 Hz
          DO 661 J=1,IUHT
  661       IF(DSTOR(2,J).LT.BND) KB=KB+1
          KBND(I)=KB-KBLL
          KBLL=KBL
          KBL=KB
          BND=BND+5.
          IF (I.GE.3 .AND.
&           KBND(I).GT.KBND(IBND)) IBND=I
  662   CONTINUE
        IF(KBND(IBND).GE.3) THEN  ;if search succeeded (>=3 pts)
          SIGLEV=(IBND+2)*5       ;shift gate to center of that band,
          GO TO 664               ;and go to tracking gate process.
        END IF                    ;if search failed (less than 3 pts),
        TEST=0.                   ;  pnt is assumed to be ref or noise
        DO 665 K=1,NAVG
          TEST=TEST+DSTOR(2,K)
  665   CONTINUE
        TEST=TEST/NAVG            ;mean freq of bottom half of DSTOR
        IF(TEST.LT.170..OR.       ;if mean falls outside ref range,
&         TEST.GT.200.) THEN      ;it is noise. In this case:
          DO 663 K=1,NAVG
            DSTOR(2,K)= DINERT    ;replace with alternating inert values
  663     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        ;if > 2 pts in gate,signal in gate
        DSUBST=SUMGTE/NGATE       ;mean freq of pts in gate
        LOS=0                     ;reset noise point counter
        DO 673 J=1,IUHT                          ;substitute mean freq
          IF(INGATE(J).EQ.0) DSTOR(2,J)=DSUBST ;for pts outside gate.
  673   CONTINUE
        SIGLEV=(SIGLEV+DSUBST)*0.5 ;adjust gate center freq
      END IF
C-------------------------------------------------------------
C                 AVERAGE RAW DATA TO OBTAIN EDITED DATA POINT
C-------------------------------------------------------------
C                 The set of .1 second raw data points in the
C                 bottom half of DSTOR will be averaged to obtain
C                 a single edited data point, EDATA. The number of
C                 raw points averaged is NAVG (was 5 in METROC)
C                 so the sampling period of the edited points is
C                 NAVG * 0.1 second (was .5 sec in METROC)
C-------------------------------------------------------------
   69 CONTINUE
      JK = JK+1                   ;index to the edited data pts
      DO 700 J=1,4                ;zero the sum array
      SUM(J) = 0.
  700 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 (IO1,1040)           ;'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 classi-
C                                   ;  -fication of signal
C-------------------------------------------------------------
C                DATA IS THERMISTOR SIGNAL
C-------------------------------------------------------------
      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)
&       EDATA(2).LT.170.) .OR.  ;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
C                               ;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 5pts
            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-------------------------------------------------------------
C                 REPLACE REFERENCE FREQUENCIES WITH 'INERT' VALUE
C-------------------------------------------------------------
C                 At this point, 5 (NAFTR0) signal points have been
C                 recieved since the last reference point, so
C                 the reference interval is assumed ended.
C                 Subroutine REWRT moves back a given number of
C                 points in the queue and replaces the frequency
C                 datum of those points with the "inert" value
C                 DINERT. The number of points rewritten, INTRVL,
C                 extends from about the 5th point preceding
C                 the reference interval to the 4th point following
C                 it. The overlap is to mask any switching transients.
C
C          Number of data points to rewrite (INTRVL) =
C          length of reference interval (JR)
C          - noise pts since last ref interval(ISKREF)
C          + 2 'verify' intervals,
C              one on each side of ref interval(2*NAFTR0)
C          - 3 points.
C-------------------------------------------------------------
        INTRVL=JR-ISKREF+2*NAFTR0-3 ;nbr of edited pts to rewrite
        CALL REWRT (INTRVL,DINERT) ;replace freqs
        JR = 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-------------------------------------------------------------
      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 --
C                 process cannot start in reference. It should
C                 be noted that this can happen only at the
C                 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
      END IF
C-------------------------------------------------------------
C                 DATA IS NOISE
```

```
C-------------------------------------------------------------------
      IF(EDATA(2).LT.10. .OR.    ;if freq is outside data and
&     EDATA(2).GT.200.) THEN     ;ref range(10-200Hz),it is noise.
        IFLAG = INOISE
        EDATA(2)=DINERT          ;replace with 'inert' value
        ISKREF= ISKREF+1         ;noise counter used in
      END IF
      IF (EDATA(2).LT.10. .OR.   ;if freq is outside(10-210Hz),
&         EDATA(2).GT.210.) IFRCK=IFRCK+1  ;increment noise cntr.
C
C-------------------------------------------------------------------
C                CONVERT TO RECTANGULAR COORDINATES
C-------------------------------------------------------------------
C               'RDRCRD' converts the azimuth, elevation, and range
C                coordinates in EDATA(3), EDATA(4), EDATA(5) into
C                rectangular X,Y,Z coordinates, returning the three
C                values back in EDATA(3), EDATA(4), EDATA(5),
C                respectively.
C-------------------------------------------------------------------
  150 CONTINUE
      CALL RDRCRD (EDATA(3),EDATA(4),EDATA(5))
C
C-------------------------------------------------------------------
C                FIND APOGEE
C-------------------------------------------------------------------
      IF (EDATA(5).LT.ZAPOG) THEN ;current altitude above stored apogee?
        ZAPOG = EDATA(5)         ;then save new apogee
        TAPOG = EDATA(1)         ;time of apogee
      END IF
C
C-------------------------------------------------------------------
C                ADD EDITED DATA POINT TO QUEUE
C-------------------------------------------------------------------
      CALL ADD1 (EDATA,FULL)     ;attempt to add point to queue1
      IF (FULL.GT.0.) THEN       ;if queue is full
        WRITE (IO1,1010)         ;'edited data queue full'
 1010   FORMAT ('EDITED DATA QUEUE IS FULL')
        GO TO 200                ;terminate real time processing
      END IF
      GO TO 60                   ;loop and get another 5
C                                ;raw data points
C-------------------------------------------------------------------
C                END OF REAL TIME PROCESSING
C-------------------------------------------------------------------
  200 CONTINUE                   ;normal exit from real time loop
      WRITE (IO1,1000)          ;'end of real time processing'
 1000 FORMAT ('END OF REAL TIME PROCESSING')
      IF (IFRCK.GT.1000 .AND.    ;more than 1000 bad data pts?
&       .NOT.IWIND)    THEN
        WRITE (IO1,1030)         ;'converting to wind-only mode'
 1030   FORMAT ('*** MORE THAN 1000 BAD FREQ. DATA POINTS FOUND'/
&       'CONVERTING TO WINDS-ONLY MODE')
        IWIND=1
      END IF
      IF (IWIND) RETURN    ;if winds-only, skip least squares
C
C-------------------------------------------------------------------
C                CALCULATE LEAST SQUARES COEFFICIENTS
```

```
C-------------------------------------------------------------------
      IF (JRP.LT.3) THEN          ;less than 3 ref intervals?
        WRITE (IO1,1020)          ;display error message
 1020   FORMAT ('*** 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(ICNTP2,LTHP1)
C
C     SUBROUTINE FITON LOADS UP A MIDPT ARRAY OF TIMES AND POSITION,
C     CALCULATES THE VELOCITIES IN TERMS OF RAW POSITION AND SMOOTHING
C     COEFFICIENTS AND STORES THESE IN ARRAYS.  ITS ARGUMENTS ARE:
C     ICNTP2 = INDEX INTO VELOCITY AND MIDPT ARRAYS WHERE DATA IS TO
C              BE STORED.
C     LTHP1  = SMOOTHING INTERVAL LENGTH FOR CALCULATING VELOCITY.
C
*
      INCLUDE 'RCOMMON.SA:1'
*
       PX=0.0
       PY=0.0
       PZ=0.
       DO 1 I=1,LTHP1
      PZX=PZX+Z(I)*PXY1(I)
      PX=PX+X(I)*PXY1(I)
    1 PY=PY+Y(I)*PXY1(I)
    5 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)
C=====================================================================
C   PURPOSE
C     Calculates the sensor temperature (TEMP) given a signal
C     frequency (FREQ).  The ratio between FREQ and the reference
C     signal (calculated by a quadratic equation for time) is
C     the search key for the table FCSOND. The sensor resistance
C     RWNT is found by interpolation from table RCSOND. A
C     second search of table RCBEAD gives the temperature.
C     The search technique is binary.
C
C   DUMMY VARIABLES
C     FAIL        Search-fail flag. FAIL=1 indicates that either
C                 FRATIO is outside the FCSOND table or RWNT is
C                 outside of the RCBEAD table so sensor temperature
C                 cannot be found.  Otherwise FAIL=-1.
C     FREQ        Sonde sensor signal frequency (Hertz) to be
C                 used to calculate sensor temperature
C     T           Time after launch (seconds).
C     TEMP        Sensor temperature (degrees Kelvin)
C
C   INTERNAL VARIABLES
C     IH,IL,IM    Upper, lower, and mid indices used in
C                 table searches
C     RATIO       Temporary storage variable
C     RWNT        Sensor resistance calculated by linear
C                 interpolation.
C
C   COMMON VARIABLES
C     C(3)        Coefficients for quadratic least squares fit to
C                 the reference signal
C     FCSOND(25)  Sonde calibration data, frequency ratio associated
C                 with corresponding sensor resistance values in
C                 RCSOND.
C     FRATIO      Ratio of sonde frequency to reference frequency
C     NCBEAD      Number of calibration points for the thermistor
C                 given for the present flight.
C     NCSOND      Number of calibration points for the sonde for
C                 the present flight.
C     RCBEAD(10)  Thermistor calibration data, resistance values
C                 corresponding to thermistor temperatures
C                 in TCBEAD.
C     RCSOND(25)  Sonde calibration data, sensor resistance values
C                 corresponding to frequency ratios in FCSOND.
C     TCBEAD(10)  Thermistor calibration data, temperature (degrees C)
C                 corresponding to resistance values in RCBEAD.
C
C   CALLED BY - DRIVER
C
C   SUBROUTINES CALLED - NONE
C
C   TERMINATION CONDITIONS
C     If no values are found in FCSOND bracketing FRATIO, or
C     no values are found in RCBEAD bracketing RWNT, TEMP
C     is left undefined, the 'search-fail' flag is set to 1,
C     and the routine returns. Otherwise, TEMP is calculated.
C
C   CHANGES
```

```
C     This routine is identical to the METROC-K version
C     except:
C     1.The flag FAIL was added to indicate search failure
C       as the RETURN1 statement is not available on this
C       machine.
C     2.The storage variable RATIO was added because AMOD
C       will not take arithmetic expressions as arguments.
C     3.The dummy variable TIME was changed to T to avoid
C       conflict with a common variable.
C   COMPILED 8/23/83
C=======================================================================
      INCLUDE 'RCOMMON.SA:1'
C
C                THE DATA IN THE 'CALBRA' LABEL COMMON ARE READ IN BY THE
C                PROGRAM 'MAIN' AND THEN ADJUSTED BY THE SUBROUTINE
C                'CLBSET',
C
      FAIL = -1.
      FRATIO = T*(T*C(1) + C(2)) + C(3)
      FRATIO = FREQ / FRATIO
C
C                SEARCH 'FCSOND' FOR VALUES BRACKETING 'FRATIO'
C
      IM = NCSOND + 1
      IL = 0
  100 IH = IM
  110 IM = IL + (IH-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
C
C                'RWNT' IS THE SONDE RESISTANCE CALCULATED BY LINEAR
C                INTERPOLATION.
C
  140 RWNT = RCSOND(IM) + (FRATIO = FCSOND(IM)) *
&            (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)
C
C                SEARCH 'RCBEAD' FOR VALUES BRACKETING RWNT
C
      IM = NCBEAD + 1
      IL = 0
  200 IH = IM
  210 IM = IL + (IH - 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
C
C                'TEMP' IS CALCULATED BY AN INTERPOLATION SCHEME
```

```
C                 USING 'TCBEAD' AND 'RCBEAD',
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
C                 SEARCH HAS FAILED
C
  500 FAIL = 1.
      RETURN
      END
```

```
      FUNCTION GPTOM (Z)
C=====================================================================
C     Converts geo-potential altitude to meters altitude
C
C   CONSTANTS
C     XX        Equatorial radius squared in meters
C     ZZ        Polar radius squared in meters
C     9.780356 Equatorial gravity constant in Lambert's Gravity Form.
C     9.8       One geo-potential unit = 9.8 geometric units
C     5.7704E-5     Geodetic constant in Lambert's Gravity Formula
C     5.1723413E-2 Geodetic constant in Lambert's Gravity Formula
C
C   DUMMY VARIABLES
C     Z         Geo-potential altitude
C     GPTOM     Altitude in meters
C
C   INTERNAL VARIABLES
C     IPASS     Flag indicating the first time the function is called
C               so that radius and gravity can be calculated. Values:
C                   1=first call of the function
C                   0=subsequent call
C     GPHE      Lambert's Gravity Formula with altitude terms omitted
C     RADUS     Computed nominal earth radius in meters
C     SN        Sine of the geodetic latitude of the radar site
C     SN2       Sine of twice the geodetic latitude of the radar site
C     LAT2      Twice the geodetic latitude of launch site in degrees
C
C   COMMON VARIABLES
C     XLAT      Geodetic latitude of the launch site in degrees
C
C   CALLING ROUTINES - DRIVER
C
C   SUBROUTINES CALLED - ZSIN, ZCOS
C
C   REFERENCES
C     F.D. Stacey, 'Physics of the earth', John Wiley and Sons,
C     1969  pp. 46-48
C
C   CHANGES
C     Identical to METROC-K version except:
C     1.Deleted unused constant GRAV
C     2.Added temporary storage variable LAT2.
C     3.Used new functions ZCOS and ZSIN because
C       COS and SIN cannot take negative arguments
C   COMPILED 8/22/83
C=====================================================================
      DATA IPASS/1/
      DATA XX/ 4.06801E13/, ZZ/ 4.04068E13/
C
C                 ON FIRST CALL COMPUTE RADIUS AND
C                 GRAVITY AT STATION LATITUDE
C
      IF (IPASS.EQ.0) GO TO 10
      IPASS = 0
      RADUS = XX*ZZ/(XX*ZSIN(XLAT)**2 + ZZ*ZCOS(XLAT)**2)
      RADUS = SQRT (RADUS)
      LAT2 = 2. * XLAT
      SN2 = ZSIN(LAT2)
```

```
      SN = ZSIN(XLAT)
      GPHE = 9.780356 + 5.1723413E-02 * SN * SN
&            -5.7704E-05 * SN2 * SN2
C
C                    CONVERT GEOPOTENTIAL ALTITUDE TO METERS
C
   10 GPTOM = (RADUS*9.8*Z) / (GPHE*RADUS - 9.8*Z)
      RETURN
      END
```

```
      FUNCTION GRVITY (Z)
C=====================================================================
C   PURPOSE
C     Computes gravity as a function of altitude Z and
C     latitude of the observer XLAT.
C
C   DUMMY VARIABLES
C     Z       Altitude in kilometers
C
C   LOCAL VARIABLES
C     GA      Computed quadratic coefficient in gravity
C             formula
C     GB      Computed linear coefficient in gravity formula
C     GRVITY Acceleration of gravity as a function of
C             altitude and latitude (meters/second squared)
C     GRAV    Nominal acceleration of gravity.
C     IPASS   Flag indicating the first time the function
C             is called. Possible values:
C               1= first call, calculate constants
C               0= subsequent call, constants already calculated
C     RADSQ   Square of RADUS, temporary variable
C     RADUS   Nominal earth radius (meters)
C     XX      Equatorial radius squared in meters
C     ZZ      Polar radius squared in meters
C
C   COMMON VARIABLES
C     XLAT    Geodetic latitude of launch site in degrees
C
C   CALLED BY - WIND, DRIVER
C
C   CHANGES
C     identical to the METROC-K subroutine except:
C     1.Variable RADSQ has been added to store intermediate
C       results (6/24/83)
C   COMPILED   6/24/83
C=====================================================================
      INCLUDE 'RCOMMON.SA:1'
      DATA IPASS /1/
      DATA GRAV / 9.79976 /,XX / 4.06801E13 /,ZZ /4.04068E13/
      IF (IPASS.EQ.0) GO TO 10
      IPASS = 0
      RADSQ = XX*ZZ/(XX*SIN(XLAT)**2 + ZZ*COS(XLAT)**2)
      RADUS = SQRT (RADSQ)
      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-----------------------------------------------------------------
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.
C
C   DUMMY VARIABLES
C     CK1,CK2,
C     CK3,CK4   Interpolated Krumins coefficients
C     OUT       Out of range flag. Values:
C                 1.=altitude is outside range of Krumins table
C                -1.=in range, coefficients returned
C     TIME      Time since launch (seconds)
C     Z         Altitude (Km) of sonde
C
C   INTERNAL VARIABLES
C     IH,IL,IM  Upper, lower, and mid index in binary table search
C     IX        Second index of CC4. 1=day, 2=night
C     RATE      Ratio of altitudes for linear interpolation
C
C   COMMON VARIABLES
C     CC1(60)   Krumins coefficient K1, aerodynamic heating factor
C               (coefficient of the square of airspeed)
C     CC2(60)   Krumins coefficient K2 (seconds), dynamic lag factor
C               (coefficient of rate of change of sensor temperature)
C     CC3(60)   Krumins coefficient K3, radiation emission factor
C               (coefficient of fourth power of sensor temperature)
C     CC4(60,2) Krumins coefficient K4 (degrees K), radiation
C               (solar, longwave) and electric heating input term
C                  CC4(j,1)=for day
C                  CC4(j,2)=for night
C     CCZ(60)   Tabulated altitudes (meters) of the Krumins
C               coefficients CC1,CC2,CC3,CC4
C     DAYNGT    Day-night flag. Values: 1.=sonde is in daylight
C                                       2.=night
C     NKMN      Number of altitude levels in the Krumins table
C
C   CALLED BY - KMNTMP, DRIVER
C
C   SUBROUTINE CALLED - SUNUP
C
C   TERMINATION CONDITIONS
C     If altitude is outside the Krumins table range, the
C     lookup fails and the routine returns with OUT=1.
C
C   CHANGES
C     Identical to METROC-K version except:
C     1.The flag OUT was added to indicate the out-of-range
C       condition.
C   COMPILED 8/20/83
C-----------------------------------------------------------------
      INCLUDE 'RCOMMON.SA:1'
C     OUT = -1.                          ;clear out-of-range flag
C
C                 THE SEARCH TECHNIQUE IS BINARY
C
      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
C
C                  INTERPOLATION
C
   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
C
C                  OUT OF RANGE
C
  100 OUT = 1.
      RETURN
      END
```

```
      SUBROUTINE KMNTMP (TAIR, CAERO, CDYN, CEMISS, CABSRB,AIRSPD)
C=======================================================
C     Calculates the corrected air temperature TAIR
C     based upon the Krumins' tables.
C
C   DUMMY VARIABLES
C     AIRSPD    Relative airspeed
C     CABSRB    Krumins temperature correction (degrees kelvin)
C               for absorbed radiation (solar and longwave)
C     CAERO     Krumins temperature correction for aerodynamic
C               heating
C     CDYN      Krumins temperature correction for
C               dynamic lag
C     CEMISS    Krumins temperature correction for
C               emitted radiation
C     TAIR      Air temperature based on Krumins corrected
C               sensor temperature (degrees Kelvin)
C
C   INTERNAL VARIABLES
C     CK1,CK2,
C     CK3,CK4 Interpolated Krumins coefficients
C     OUT       Out-of-range flag. Values:
C                 1.=altitude ZS is outside range of Krumins table
C                 0.=in range
C
C   COMMON VARIABLES
C     TEMP      Sensor temperature (degrees Kelvin)
C     TEMPD     Rate of change of sensor temperature (degrees K/sec)
C     TIME      Elapsed time since launch (seconds)
C     ZS        Altitude of sonde (meters)
C
C   CALLED BY - DRIVER
C
C   SUBROUTINES CALLED - KMNTAB
C
C   CHANGES
C     Identical to METROC-K version except:
C     1.Flag OUT added to indicate out-of-range condition
C       in KMNTAB.
C   COMPILED 8/22/83
C=======================================================
      INCLUDE 'RCOMMON.SA:1'
      CALL KMNTAB (ZS, CK1, CK2, CK3, CK4, TIME, OUT)
      IF (OUT)  GO TO 10
      CAERO = -(CK1 * AIRSPD**2)
      CDYN = CK2 * TEMPD
      CEMISS = CK3 * TEMP**4
      CABSRB = -CK4
      TAIR = TEMP + CAERO + CDYN + CEMISS + CABSRB
      RETURN
C
C * OUTSIDE KRUMINS TABLE RANGE, NO TEMP COR APPLIED ....
   10 CAERO=999.9
      CDYN=999.9
      CEMISS=999.9
      CABSRB=999.9
      TAIR=TEMP
      RETURN
```

```
        END
```

```
      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(10,11),EI(10),CON1(10,10),CON2(10,10)
      DATA  BINOM/1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,
     &    -2.,-6.,-12.,-20.,-30.,-42.,-56.,-72.,-90.,-110.,
     &    0.,6.,30.,90.,210.,420.,756.,1260.,1980.,2970.,
     &    0.,0.,-20.,-140.,-560.,-1680.,-4200.,-9240.,-18480.,-34320.,
     &    0.,0.,0.,70.,630.,3150.,11550.,34650.,90090.,210210.,
     &    0.,0.,0.,0.,-252.,-2772.,-16632.,-72072.,-252252.,-756756.,
     &    0.,0.,0.,0.,0.,924.,12012.,84084.,420420.,1681680.,
     &    0.,0.,0.,0.,0.,0.,-3432.,-51480.,-411840.,-2333760.,
     &    0.,0.,0.,0.,0.,0.,0.,12870.,218790.,1969110.,
     &    0.,0.,0.,0.,0.,0.,0.,0.,-48620.,-923780.,
     &    0.,0.,0.,0.,0.,0.,0.,0.,0.,184756./
      DATA  CON1/0.,-1.,2.,-6.,24.,-120.,720.,-5040.,40320.,-362880.,
     &    0.,2.,-6.,22.,-100.,548.,-3528.,26136.,-219168.,2053152.,
     &    0.,0.,3.,-18.,105.,-675.,4872.,-39396.,354372.,-3518100.,
     &    0.,0.,0.,4.,-40.,340.,-2940.,27076.,-269136.,2894720.,
     &    0.,0.,0.,0.,5.,-75.,875.,-9800.,112245.,-1346625.,
     &    0.,0.,0.,0.,0.,6.,-126.,1932.,-27216.,379638.,
     &    0.,0.,0.,0.,0.,0.,7.,-196.,3822.,-66150.,
     &    0.,0.,0.,0.,0.,0.,0.,8.,-288.,6960.,
     &    0.,0.,0.,0.,0.,0.,0.,0.,9.,-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.,708744.,-7036200.,
     &    0.,0.,0.,12.,-120.,1020.,-8820.,81228.,-807408.,8684160.,
     &    0.,0.,0.,0.,20.,-300.,3500.,-39200.,448980.,-5386500.,
     &    0.,0.,0.,0.,0.,30.,-630.,9660.,-136080.,1898190.,
     &    0.,0.,0.,0.,0.,0.,42.,-1176.,22932.,-396900.,
     &    0.,0.,0.,0.,0.,0.,0.,56.,-2016.,48720.,
     &    0.,0.,0.,0.,0.,0.,0.,0.,72.,-3240.,
     &    0.,0.,0.,0.,0.,0.,0.,0.,0.,90.,
     &    0.,0.,0.,0.,0.,0.,0.,0.,0.,0./
  100 FORMAT(14,11F11.7)
      N=NPTS-1
      MPM1=MPWR-1
C
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
C        COMPUTE BASIC CONSTANTS FOR COEFFICIENTS:
C        DM(K)=N(N-1) (N-2)...(N+1-K)
C        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
```

```
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
        DO 8 I=1,MID
        DO 6 M=1,MPWR
        C(M)=0.
        LIM=M+1
        DO 5 K=1,LIM
        C(M)=C(M)+EM(K,I)*BINOM(M,K)
        TEMP=C(M)
        IF(ABS(TEMP).LT. .1E-15)  C(M) = 0.
      5 CONTINUE
        CF(M,I)=C(M)
      6 CONTINUE
C
C         GENERATE SECOND HALF OF COEFFICIENTS
        INDX=N+2-I
        DO 7 M=1,MPWR,2
      7 CF(M,INDX)=-CF(M,I)
        DO 8 M=2,MPWR,2
      8 CF(M,INDX)=CF(M,I)
C
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
C         EI(M)=I**M
        EII=I-1
        EI(1)=EII
        DO 20 M=2,MPWR
     20 EI(M)=EII*EI(M-1)
C             EM(M,I)=NCP(M) OF I
        EM(1,I)=1.
        DO 21 M=2,MPWR
        EM(M,I)=CON1(M,1)
        DO 21 K=2,M
     21 EM(M,I)=EM(M,I)+CON1(M,K)*EI(K-1)
C         EM(M,I)=NCP(M)/DM(M+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 CF1(M,I)=C(M)
```

```
C
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
           EM(M,I)=CON2(M,1)
           DO 31 K=2,M
        31 EM(M,I)=EM(M,I)+CON2(M,K)*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
C          COMPUTE SECOND DERIVATIVE COEFFICIENTS (FIRST HALF)
           DO 34 M=1,MPWR
           C(M)=0.
           DO 33 K=2,M
           C(M)=C(M)+EM(K,I)*BINOM(M,K+1)
           TEMP=D(M)
           IF(ABS(TEMP).LT. .1E-15)  C(M)=1.
        33 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
```

```
*BEGIN                  112395
*-----------------------------------------------------------------
*     9/28/82
*     CHANGES MADE:
*        NONE
*-----------------------------------------------------------------
CMAIN    MAIN PROGRAM FOR 'K' VERSION OF METROC SYSTEM OF PROGRAMS
C
C  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C    * NAME   - METROC-K
C
C    * NUMBER - 112395
C
C    * ABSTRACT - METROC-IK IS DESIGNED FOR TOTAL AUTOMATED DATA
C        REDUCTION OF METEOROLOGICAL ROCKETSONDE PARAMETERS.  THE ONLY
C        MANUAL INTERFACE REQUIREMENT IS THE INPUT OF SONDE CALIBRATION
C        DATA, SUPPORTING RAWINSONDE DATA, AND NECESSARY START AND STOP
C        INFORMATION FOR THE INPUT TAPE.  THE PROGRAM USES A METPASS 1
C        (1,1,2397) OUTPUT TAPE.  THE DATA ARE EDITED IN A CLEAN-UP
C        ACTIVITY AND PASSED THRU ADDITIONAL SMOOTHING AND FILTERING
C        WHERE WIND, TEMPERATURE, PRESSURE, AND OTHER PARAMETERS ARE
C        CALCULATED.  THE PROGRAM ALSO AVERAGES THE VELOCITY COMPONENTS
C        USING A 21 POINT AVERAGING SCHEME.  THE DATA ARE REFORMATTED
C        AND A BASELINE TIE-ON LEVEL (BETWEEN RADIOSONDE AND ROCKET
C        DATA) IS FOUND.
C
C    * PURPOSE
C        ALL INPUT CARDS FOR METROC-K ARE READ IN THIS ROUTINE AND
C        DISPLAYED FOR EASY VERIFICATION.  SUBROUTINE EDIT IS CALLED TO
C        PASS ONCE THROUGH THE APPARATUS: DATA, CHECK FOR VALIDITY, AND
C        EDIT IT.  DRIVER IS CALLED TO PASS THROUGH THE DATA AGAIN,
C        CALCULATING METEOROLOGICAL DATA.  METFORM IS CALLED TO AVERAGE
C        VELOCITY CAMPONENTS AND DETERMINE THE TIE-ON LEVEL.
C
C    * COMMON BLOCKS - /CTIM/, /IFMOP/, /TEMPA/, /CDATA/, /CGATE/,
C                      /CALBRA/
C
C    * INPUTS
C        BPRES  - 30 WORD ARRAY OF RADIOSONDE PRESSSURES
C        BTMP   - 30 WORD ARRAY OF RADIOSONDE TEMPERATURE (CELSIUS)
C        CALD   - SODE SENSOR FREQUENCY CALIBRATION DATUM
C        DPP    - 30 WORD ARRAY OF DEWPOINT
C        HEADER - 14 WORD LAUNCH OPERATION LABEL
C        HRADAR - RADAR SITE HEIGHT ABOVE SEA LEVEL IN METERS
C        ICAL   - NOT USED IN METROC-K
C        IENT   - SWITCH, 1 = CALIBRATION REQUIRED
C        IFILE  - FILE NUMBER FOR USE WITH MULTIFILE INPUT TAPES
C        IFM    - OPTION, 0 = CHANGE SLANT RANGE FROM FEET TO METERS
C        IRSKIP - NUMBER OF INPUT RECORDS TO SKIP BEFORE PROCESSING
C        IWIND  - LOGICAL SWITCH, 1. = RADAR DATA ONLY, -1. = TEMP.
C        IZTIME - LAUNCH TIME IN HOURS, MINUTES AND SECONDS (INTEGER)
C        LALT   - 30 WORD ARRAY OF RADIOSONDE ALTITUDE IN DECAMETERS
C        LDIR   - 30 WORD ARRAY OF RADIOSONDE WIND DIRECTION IN DEGREES
C        LSPD   - 30 WORD ARRAY OF RADIOSONDE WIND SPEED (M/S)
C        NCBEAD - NUMBER OF THERMISTOR CALIBRATION POINTS
C        NCSOND - NUMBER OF SONDE CALIBRATION POINTS
C        NPBL   - NUMBER OF RADIOSONDE DATA POINTS
C        RCBEAD - CALIBRATION DATA FOR SENSOR
```

```
C         RCSOND - SONDE CALIBRATION DATA
C         REF    - SONDE REFERENCED FREQUENCY PERIOD IN SECONDS
C         TCBEAD - THERMISTOR CALIBRATION DATA  (TEMPERATURE DEG. CELSIUS)
C         TSTART - TIME TO START PROCESSING DATA (SECONDS AFTER LAUNCH)
C         TSTOP  - TIME TO STOP PROCESSING DATA (SECONDS AFTER LAUNCH)
C         XLAT   - GEODETIC LATITUDE OF LAUNCH SITE IN DEGREES
C         XLONG  - GEODETIC LONGITUDE OF LAUNCH SITE IN DEGREES
C
C     * OUTPUT
C         HEADER, IZTIME, TSTART, TSTOP, IRSKIP, HRADAR, XLAT, XLONG,
C         IWIND, RCSOND, FCSOND, REF, CALD, TCBEAD, RCBEAD - CARD IPUT
C                  DATA PRINTED ON OUTPUT LISTING
C         HGATE  - HALF-WIDTH OF SIGNAL GATE USED FOR TRACKING MET DATA
C         SIGLEV - INITIAL VALUE OF SIGNAL GATE USED TO ACQUIRE, FILTER,
C                  AND TRACK MET DATA = 155.0
C         TZERO  - GMT LAUNCH TIME IN SECONDS
C
C     * INTERNAL VARIABLES
C         BI     - OCTAL VALUE FOR ALL BLANKS 0377777777777
C         DTOR   - CONVERSION FACTOR FOR DEGREES TO RADIANS
C         MXBEAD - MAXIMUM NUMBER OF THERMISTOR SENSOR CALIBRATION POINTS
C         MXSOND - MAXIMUM NUMBER OF SONDE CALIBRATION POINTS
C
C     * INPJT FILES
C         05       CARD       BCD       METROC-K INPUT CARDS 1 TO 37
C
C     * OUTPUT FILES
C        06       LISTING    BCD       METROC-K INITIALIZATION AND
C                                        CALIBRATION DATA
C     * SCRATCH FILES - NONE
C
C     * CALLING ROUTINES - NONE
C
C     * SUBROUTINES CALLED - ERROR, CLBSET, EDIT, LDEDIT, DRIVER, METFORM
C
C     * COMMENTS
C         IF CALIBRATION DATA IS NOT WITHIN BOUNDS OR FCSOND AND RCBEAD
C         VALUES ARE MOT INCREASING, AN ERROR MESSAGE IS PRINTED AND JOB
C         IS TERMINATED
C
C     * MODIFICATION HISTORY
C         PROGRAMMER            DATE     MODIFICATION
C          BILL SPEIDEL        9/02/80  *ADDED FILE SKIP OPTION (FC 21)
C                              2/11/81  *READ FEET TO METERS CONVERSION
C                                        OPTION FROM INPUT CARD 4
C                                       *PRINT CARD 4 ON FILE 06
C          CHARLOTTE TETER     8/24/81  *CALL METFORM
C
C  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C//
C
        COMMON /CTIM/ ICAL,IENT,IDN
      COMMON /IFMOP/ IFM
        COMMON /TEMPA/ DRTEMP(100),DRTIM(100),NOTEMP,NTPT
      COMMON /CDATA/ HEADER(14),TSTART,TZERO,TSTOP,IWIND,
     &  HRADAR, XLAT, XLONG,
     &  NFBL,LALT(30),LSPD(30),LDIR(30),DPP(30),BTMP(30),BPRES(30),ALTG,
     &  IRSKIP,LEV
```

```
      COMMON /CGATE/ SIGLEV,HGATE
      COMMON /CALBRA/ FCSOND(25), RCSOND(25), TCBEAD(10), RCBEAD(10),
&      NCBEAD, NCSOND
      DIMENSION IZTIME(3)
      DATA MXBEAD /10/, MXSOND /25/
        DATA BI/0377777777777/
      DATA DTOR / .01745329/
      DATA IFC /21/
C
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
C         * PERFORM FILE SKIP
      REWIND (21)
      IF (NFILE.LE.0) GO TO 5
      CALL FILFSP (IFC,NFILE)
    5 CONTINUE
C
      XLAT = XLAT * DTOR
      XLONG = XLONG * DTOR
        IF(IWIND) GO TO 16
C                 CONVERT DEGREES TO RADIANS
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,NCBEAD
C
C+    *PRINT ERROR MESSAGE IF CALIBRATION DATA IS OUT OF BOUNDS OR FCSOND
C+       AND RCBEAD VALUES NOT INCREASING
      IF (NCBEAD.GT.MXBEAD) CALL ERROR(4)
      IF (NCSOND.GT.MXSOND) CALL ERROR(5)
      IF (NCBEAD.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
```

```
       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) NPBL,(LALT(I),LSPD(I),LDIR(I),DPP(I),BTMP(I),BPRES(I)
&     ,I=1,NPBL)
      IF (IWIND) GO TO 30
C
C+    *MODIFY CALIBRATION DATA
C               CLBSET'MODIFIES THE TCBEAD, RCBEAD, RCSOND, AND FCSOND
C               ARRAYS FOR LATER USE BY THE SUBROUTINE 'FRQTMP'.
      CALL CLBSET
C
C
C+    *EDIT, SMOOTH AND FILTER METEOROLOGICAL ROCKETSONDE PARAMETERS
   30 CALL EDIT
      CALL UDEDIT(IWIND)
      CALL DRIVER
      CALL METFORM
      STOP
C
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, 14A6 //
&         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-----------------------------------------------------------
C   PURPOSE
C     Calculates the coefficients for a polynomial
C     least-squares fit by solving the matrix equation
C              AC = B
C     The elements of A are supplied by array BT,
C     which is generated by EDIT along with array B.
C     C is the solution vector. The technique used
C     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
C     B(i)     = Sum(freq*time**(N-i))
C
C     BT(i)    =
C
C
C   DUMMY VARIABLES
C     N       Number of coefficients in the fitted polynomial,
C             equals one plus the degree of the polynomial
C
C   LOCAL VARIABLES
C     DL
C     I
C     IL
C     IXA
C     J
C     JX
C
C   COMMON VARIABLES
C     A(3,3)
C     B(3)
C     BT(5)
C     C(3)
C
C   CHANGES
C     Identical to the METROC-K version except:
C     1.Array A in the origional routine, which was
C       equivalent to common array BT, was replaced
C       with BT.
C     2.Array U was renamed A.
C
C   REFERENCES
C     Equations for coefficients: F.L. Staffanson, "METROC-K
C     Algorithms", University of Utah Engineering College
C     Report UTECMR 79-161 November 1979, pp. 16-17.
C-----------------------------------------------------------
      INCLUDE 'RCOMMON.SA:1'
C-----------------------------------------------------------
C                 ELIMINATION
C-----------------------------------------------------------
      IL = 0
      DO 30 I = 1,N
      IXA = 2*(N-IL)
      DO 10 J = I,N                 ;transfer Ith row into matrix A
```

```
      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                                         to get multiplier DL
      B(I) = B(I)-DL*B(J)
      DO 20 JX = I,N                     ;subtract DL times Ith row
   20 A(I,JX) = A(I,JX)-DL*A(J,JX)       ;from Jth row
   30 IL = IL+1
C-------------------------------------------------------------
C                  BACK SOLUTION
C-------------------------------------------------------------
   40 I = N
   50 IF (I.EQ.IL) GO TO 60
      B(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)
C
C   DUMMY VARIABLES
C                Azimuth, elevation, and range input from EDIT
C                and X, Y, Z coordinates passed back to EDIT:
C     RDATA(1)    azimuth (degrees),    X coordinate (meters)
C     RDATA(2)    elevation (degrees), Y coordinate (meters)
C     RDATA(3)    range (meters),       Z coordinate (meters)
C
C   CONSTANTS
C     CRVETH     Reciprocal of twice the earth's radius (1/meters)
C     DTOR       Degrees to radians conversion factor
C
C   INTERNAL VARIABLES
C     AZR        Radar azimuth angle in radians
C     CA         Cosine of radar azimuth
C     CE         Cosine of radar elevation
C     ELR        Radar elevation angle in radians
C     R          Horizontal (ground) range of sonde from radar
C     SA         Sine of radar azimuth angle
C     SE         Sine of radar elevation angle
C
C   COMMON VARIABLES
C     HRADAR     Height above sea level (meters) of radar site
C
C   CALLING ROUTINE - EDIT
C
C   SUBROUTINES CALLED  -  ZSIN, ZCOS
C
C   REFERENCES
C     Conversion formula: F.L. Staffanson, "METROC-K Algorithms"
C     University of Utah Engineering College Report UTECMR 79-161
C     November 1979, pp.10-11
C
C   CHANGES
C     This subroutine is identical to METROC-K version
C     except ZSIN and ZCOS functions replace standard 6809
C     trig functions because they don't take neg.arguments
C   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
C                                   ;radius
C
      AZR = RDATA (1) * DTOR  ;convert azimuth from degrees to radians
      ELR = RDATA (2) * 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 REMOV1 (D,EMTY)
C-----------------------------------------------------------------
C   For documentation see ADD1
C-----------------------------------------------------------------
      INCLUDE 'RCOMMON.SA:1'
      DIMENSION D(5)
      EMTY=-1.
&     IF (ITAIL1.EQ.IHEAD1) THEN
        EMTY=1.
        RETURN
      END IF
      DO 10 I=1,5
        D(I)=QFILE(I,ITAIL1)
   10 CONTINUE
      ITAIL1=ITAIL1+1
      IF (ITAIL1.GT.LNGTHQ) 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
C     Replaces the frequency datum of the last INTRVL
C     edited data points in queue1 with an "inert" value,
C     DINERT.  Rewrites that would overrun the beginning
C     of the queue are stopped at the beginning (ITAIL1).
C
C                    --->         (empty)
C                  DDDDDDDDDDD_____________EEEEEEEEEEEE
C       queue 2    D           ^             ^          E  queue 1
C      (values     D         IHEAD2        ITAIL1       E (data points
C       from       D                                    E  from
C       DRIVER)    D                                    E  EDIT)
C                  DDDDDDD_______EEEEEEEEEEEEEEEEEEEEE
C                        ^       ^         <---.
C                     ITAIL2    IHEAD1
C
C   DUMMY VARIABLES
C     DINERT         Constant to replace the frequency value
C                    in QFILE(2,j) in the points to be rewritten.
C     INTRVL         Number of points in the queue to be rewritten,
C                    starting at the last point added.
C
C   INTERNAL VARIABLES
C     I              Index of points to be rewritten.
C     ISTART         Start pointer. Contains index in queue of
C                    first point to be rewritten.
C     ISTOP          Stop pointer. Contains index in queue of
C                    last point to be rewritten, which is the
C                    last point in the queue.
C
C   COMMON VARIABLES
C     IHEAD1         Head pointer for queue1. Points to location
C                    of next point to be added to queue1.
C     ITAIL1         Tail pointer for queue1. Points to location
C                    of next point to be removed from queue1.
C     QFILE(5,2000) Storage queue containing edited data points
C                    from EDIT and meteorological values from
C                    DRIVR1.
C                      QFILE(1,j)= time since launch(seconds)
C                      QFILE(2,j)= frequency datum (Hertz)
C                      QFILE(3,j)= X coordinate (meters)
C                      QFILE(4,j)= Y coordinate (meters)
C                      QFILE(5,j)= Z coordinate (meters)
C
C   CALLING ROUTINES - EDIT
C
C   COMMENTS
C     1.Subroutine ADD1 initializes the queue pointers
C       used by all the other queue access routines;
C       ADD2, REMOV1, REMOV2, REWRT. Therefore ADD1
C       must be called at least once before any of the
C       others are called.
C     2.REWRT will only work properly if IHEAD1 is
C       greater than ITAIL1. This should always be
C       true in EDIT and should be checked if the
C       initial values of the pointers are changed.
```

```
C
C    CHANGES
C      REWRT is a new routine and was not part of METROC-K.
C
C    COMPILED 6/20/82
C    REFERENCES
C-----------------------------------------------------------------
       INCLUDE 'RCOMMON.SA:1'
       IF (IHEAD1.LE.ITAIL1) THEN ;check proper pnter condition
         WRITE (IO1,1000)
         STOP
       END IF
       IF (INTRVL.LE.0) RETURN    ;handles case of INTRVL=0.
       ISTOP=IHEAD1-1             ;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
C     SUBROUTINE SINGLE CALCULATES SMOOTHING COEFFICIENTS, ITS ARGUMENTS
C     ARE:
C     IDEG1    = THE DEGREE OF TIME SMOOTHING POLYNOMIAL
C     NPTFT1   = THE SMOOTHING INTERVAL LENGTH
C     CP1      = DATA COEFFICIENTS
C     CV1      = VELOCITY COEFFICIENTS
C     DA1      = ACCELERATION COEFFICIENTS
C
      DIMENSION P1(10,99),PP1(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,P1,PSQ1,PP1,PPP1,2)
      MID=NPTFT1/2+1
      PTFT1=NPTFT1
C         COMPUTER 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/PTFT1
      CV1(I)=P1(1,I)*PP1(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)+P1(J,I)*PP1(J,MID)/PSQ1(J)
    5 CA1(I)=CA1(I)+P1(J,I)*PPP1(J,MID)/PSQ1(J)
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SUNUP (ALT,TIME)
C=====================================================================
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
C   CONSTANTS
C     DTOR        Degrees to radians conversion factor
C     364.242     Number of days in a year
C     23.4438333  Earth's inclination in degrees
C     279.9348    Factor for assymmetry due to ellipticity
C                 in computing solar declination
C     3600        Hours to seconds conversion factor
C
C   DUMMY VARIABLES
C     ALT      Altitude in meters
C     TIME     Elapsed time from launch in seconds
C
C   INTERNAL VARIABLES
C     A        The sunrise/sunset solar elevation angle (radians)
C     COSD     Cosine of the solar declination angle
C     D        Day of the year in radians
C     SIND     Sine of the solar declination angle
C     H        Solar hour angle (in decimal hours). (this is also
C              the number of hours in half a "day")
C     IYR      Year of launch decoded from header
C     MONTH    Month of launch decoded from header
C     MTHDAY   Launch day of month decoded from header
C     SNOON    The time (in decimal hours) of meridian passage
C              or true solar noon.
C     TIMNOW   Time corresponding to altitude of sensor (GMT
C              in seconds).
C     TSR      Time of sunrise (GMT in seconds).
C     TSS      Time of sunset (GMT in seconds).
C     TWODEE   Two times the day of the year in seconds
C     XPHI     Angle (in radians) which takes into account the
C              ellipticity of the earth.
C
C   COMMON VARIABLES
C     DAYNGT   Flag that indicates whether sensor is in day
C              or night. Values: 1.=day, -1.=night
C     HEADER   14 character launch operation label
C     TZERO    Launch time (GMT in seconds)
C     XLAT     Geodetic latitude of launch site in degrees
C     XLONG    Geodetic longitude of launch site in degrees
C
C   CALLED BY - KMNTAB
C
C   FUNCTIONS CALLED - ZSIN, ZCOS
C
C   COMMENTS
C     1.Date of launch must start at character 15 of HEADER
C       and be in form YYMMDD.
C     2.Longitude XLONG is in degrees counted positive
```

```
C       west of Greenwich.
C     3.All formulas for this routine were derived from
C       NASA's Technical Memorandum TM-X-1646
C
C   CHANGES
C     Identical to METROC-K version except:
C     1.Changed variable DAYMNTH to MTHDAY to make it integer,
C       and variable M to SNOON to make it real.
C       Variable DAYEAR is real instead of integer.
C     2.Used ENCODE statement to help decode HEADER
C     3.Changed the two values of flag DAYNGT to 1.=day, -1.=night
C     4.Added temporary variable TWODEE
C     5.Used functions ZSIN and ZCOS because SIN and COS cannot
C       take negative arguments.
C=================================================================
      INCLUDE 'RCOMMON.SA:1'
      DIMENSION  DAYEAR(12)
C
C                 INITIALIZE
C
      DATA DAYEAR /0.,31.,59.,90.,120.,151.,
&                  181.,212.,243.,273.,304.,334./
      DATA DTOR/.01745329/
C
C                 CALCULATE DAY OF YEAR
C
      ENCODE 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
C
C                 COMPUTE SOLAR HOUR ANGLE
C
      TWODEE = 2.* D
      XPHI = (279.9348 + D/STOR + 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(BB))/(15.*DTOR)
C
C                 COMPUTE TRUE SOLAR NOON
C
      SNOON = 12. + 0.123570 * ZSIN(D) - 0.004289 * ZCOS(D)
&           + 0.153809 * ZSIN(TWODEE) + 0.060783 * ZCOS(TWODEE)
      TIMNOW = TZERO + TIME
C
C                 DETERMINE IF DATA OBTAINED IN LIGHT OR DARKNESS
C
      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)
C-----------------------------------------------------------------
C     PURPOSE
C       Exchanges the 13 smoothed temperature and position
C       variables generated by DIRSIT with the values stored
C       in array SCOPY.
C
C     DUMMY VARIABLES
C       SCOPY(13)    contains previous values of the 13 temperature
C                    and position variables below in the order
C                    listed.
C                    SCOPY(1)=TIME ... SCOPY(13)=TEMPDD
C
C     COMMON VARIABLES
C       TIME         time after launch of present data point (s)
C       XS           smoothed east coordinate (m)
C       YS           smoothed north coordinate (m)
C       ZS           smoothed altitude (m)
C       TEMP         smoothed sensor temperature (K)
C       XD           east  velocity component (m/s)
C       YD           north velocity component (m/s)
C       ZD           upward velocity component (m/s)
C       TEMPD        first derivative of sensor temperature (K/s)
C       XDD          east acceleration component (m/s**2)
C       YDD          north acceleration component (m/s**2)
C       ZDD          upward acceleration component (m/s**2)
C       TEMPDD       second derivative of sensor temperature (K/s**2)
C
C     INTERNAL VARIABLES
C       SAV          temporary storage
C
C     SUBROUTINES CALLED - NONE
C
C     CALLING SUBROUTINES - DRIVR1
C
C     LAST CHANGE  12/6/82
C-----------------------------------------------------------------
*
      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 = TEMPDD
      TEMPDD = 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 numb   of errors in the given set of points.
C     When the maximum is exceeded, the program aborts.
C
C   DUMMY VARIABLES
C     DSTOR(5,10) tenth second raw data record.
C                    DSTOR(1,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
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
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
C   CALLED BY - EDIT
C
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
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
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
C   COMPILED  6/25/83
C-----------------------------------------------------------------
      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.              ;if difference is out of
&         DELT.LT..05)  THEN             ;    normal range,
        IERFG = IERFG + 1                ;increment error counter
        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,TXOUT,TVOUT,TAOUT,ICNTT,LTHT,
&        ISLIDE)
C
C*****************************************************************************
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 P1 AND P2 ARRAYS MUST BE DIMENSIONED AT LEAST ?LTHT?, WHILE
C     THE TEMP AND TIME ARRAYS MUST BE DIMENSIONED BY AT LEAST ?(LTHT +
C     NSKIP)? (NSKIP IS DEFINED IN SUBROUTINE DIRSIT).
C
C*****************************************************************************
C
      DIMENSION TEMP(100),P1(100),P2(100),TIME(100)
      GO TO (10,20,20,60)  IDRSIT
   10 CONTINUE
      DELT=0.5
C
C    DEFINE TIME INCREMENT
      ITBAR=(LTHT+1)/2
C
C    CALCULATE MIDPOINT OF SMOOTHING INTERVAL
      POSQR=LTHT
C
C    INITIALIZE
      P1SQR=0.0
      P2SQR=0.0
C
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,LTHT
      P1(I)=I-ITBAR
      P2(I)=(I-ITBAR)**2 - (LTHT**2-1.0)/12.0
      P1SQR=P1SQR+P1(I)**2
      P2SQR=P2SQR+P2(I)**2
    5 CONTINUE
      RETURN
   20 CONTINUE
C
C    PUT RAW TEMPERATURE AND TIME IN THE ?TEMP? AND?TIME? ARRAYS(RESP)
      TEMP(ICNTT)=T
      TIME(ICNTT)=TIN
      RETURN
   60 CONTINUE
C
C    PUT RAW TEMPERATURE AND TIME IN THE TEMP AND ARRAYS
      TEMP(ICNTT)=T
      TIME(ICNTT)=TIN
C
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,LTHT
      P0XSM=P0XSM+TEMP(I)
      P1XSM=P1XSM+(P1(I)*TEMP(I))
```

```
      P2XSM=P2XSM+(P2(I)*TEMP(I))
   80 CONTINUE
C
C    CALCULATE SMOOTHED TEMPERATURE AND ITS DERIVATIVES.
      TXOUT=P0XSM/P0SQR
      TVOUT=P1XSM/(DELT*P1SQR)
      TAOUT=2.0*P2XSM/(P2SQR*DELT**2)
C
C    SET ?T? AND ?TOUT? EQUAL TO THE RAW TEMPERATURE AND TIME (RESP)
C    CORRESPONDING TO THE SMOOTHING INTERVAL MIDPOIN.
      T=TEMP(ITBAR)
      TOUT=TIME(ITBAR)
C
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
```

```
C
      FUNCTION ZCOS(A)
C
C    THE 6809 CANT TAKE COS OF A NEG #
C
     DUMMY=ABS(A)
     ZCOS=COS(DUMMY)
     RETURN
```

```
C
      FUNCTION ZSIN(A)
C
C   THE 6809 WILL NOT TAKE THE SIN OF A NEG #
C   THIS FUNCTION TAKES CARE OF THE PROBLEM
C
     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   BT(5)
      COMMON   BTMP(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   DMMY
      COMMON   DMY(183)
      COMMON   DPP(30)
      COMMON   DRTEMP(100)
      COMMON   DRTIM(100)
      COMMON   DUM(184)
      COMMON   DUMMY(200)
      COMMON   FCSOND(25)
      COMMON   FRATIO
      COMMON   FSKRID
      COMMON   FSKT
      COMMON   HEADER(14)
      COMMON   HGATE
      COMMON   HRADAR
      COMMON   ICAL
      COMMON   IDENT
      COMMON   IDFSK
      COMMON   IDN
      COMMON   IENT
      COMMON   IFM
      COMMON   IMODEL
      COMMON   IOUT
      COMMON   IRSKIP
      COMMON   IWIND
      COMMON   LALT(30)
      COMMON   LDIR(30)
      COMMON   LEV
      COMMON   LSPD(30)
      COMMON   NAVG
      COMMON   NCBEAD
      COMMON   NCSOND
      COMMON   NKMN
      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   TEMPDD
      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   YVM(50)
      COMMON   ZAPOG
      COMMON   ZD
      COMMON   ZDD
      COMMON   ZMID(50)
      COMMON   ZS
      COMMON   ZXM(50)
```