UNITED STATES DEPARTMENT OF THE INTERIOR

GEOLOGICAL SURVEY

Transcription of Geomagnetic Variation Data From Sea Data Cassettes to Tape Using the HP9640A

by

David V. Fitterman

13 January 1981

Open-File Report 81-95

Contents

Figures

Tables

1. Introduction

Purpose

The programs described in this report were developed to transcribe geomagnetic variation data recorded on magnetic cassettes by a Sea Data Model 651-2 Data Logger to 9-track magnetic tape using a Hewlett-Packard (HP) 9640A Multiprogramming System. The data goes through several stages including being read to disk, unpacked, edited, and summarized before being written onto tape. The programs operate in an automatic mode requiring little operator intervention except for data errors, which require "human wisdom" to correct.

Appendix B is a user's guide which describes the operation of the transcription programs. Most users will need only be concerned with this section. It is not, however, entirely self-contained; so from time to time the user may have to refer to the body of this report or program listings for answers to specific questions.

Software and Hardware Requirements

Most of the transcription programs are written in HP FORTRAN IV with some subroutines and the program CASDS written in HP Assembly Language. The programs were designed to run with the HP RTE-II operating system and make use of the Spool Monitor Package (SMP, also referred to as File Manager). Using the newer RTE-IV operating system may cause some problems with program CASDS, but I have not had any direct experience with this. Some of the assembly language routines make use of special instructions which are not found on the older HP-2100 CPU. These instructions will have to be simulated if the routines are not run on an HP-21MX or newer CPU.

The software is written to run on an HP-9640A Multiprogramming System. The essential hardware is an HP-21MX CPU; an HP7900A Disk Drive; an HP7970B 9-track, 800 bpi digital tape unit; a terminal; and a Sea Data Model 12 four-track cassette reader. The cassette reader is connected to the CPU via an HP12566B Microcircuit Interface Kit. Jumper connections for the interface card are given in Table 1.1 below.

Table 1.1 HP12566B interface jumper connections

Jumper	Connection
W1	B
W2	A
W3	A
W4	B
W5	connected
W6	connected
W7	connected
W8	connected
W9	B

Use of the software with newer disk drives such as the HP7905, HP7906, or HP7920 would probably require rewriting of assembly-language disk-drive commands in program CASDS. Otherwise, the programs should work without any problems.

Program CASDS is the only software that will have to be modified to correspond to the Select Code (SC) and unit number of the disk drive and cassette reader. The listings presented in Appendix C use SC's 11B and 12B for the disk drive. The disk drive is unit number 0. The cassette reader is expected to use SC 21B. Refer to the section describing CASDS for the necessary changes for different select-code configurations.

All programs use the logical-unit number assignments given in Table 1.2.

Table 1.2 Logical-unit number assignments

LU	Name	Device
1	LUTTY	terminal
6	LUPRT	line printer
8	LUTAP	magnetic-tape drive
10	LUDSK	peripheral disk (scratch disk)

These LU assignments are set in data statements in each program so they can easily be changed at compilation time for different systems.

Each program description contains a section entitled "Program Loading". Loading is the procedure which translates the relocatable output of the FORTRAN compiler or Assembler into executable image form. Loading need only be done when the software is installed on a new system. This is not something the user typically needs to do.

System Overview

The transcription process is controlled by program TRANZ. It requests data concerning the cassette to be transcribed. Once these data are input by the operator, TRANZ schedules programs CASDS, UNPKZ, EDITZ, and DBHIZ in turn as though they were subroutines. The various programs communicate through a system-common block of 128 words. Program MGAIN is used to create and edit a magnetometer gain file named MAGAIN. Transfer files /TRANZ and \TRANZ are used to restore and shut down the transcription system before and after transcription, respectively.

2. TRANZ - Transcription Control Program

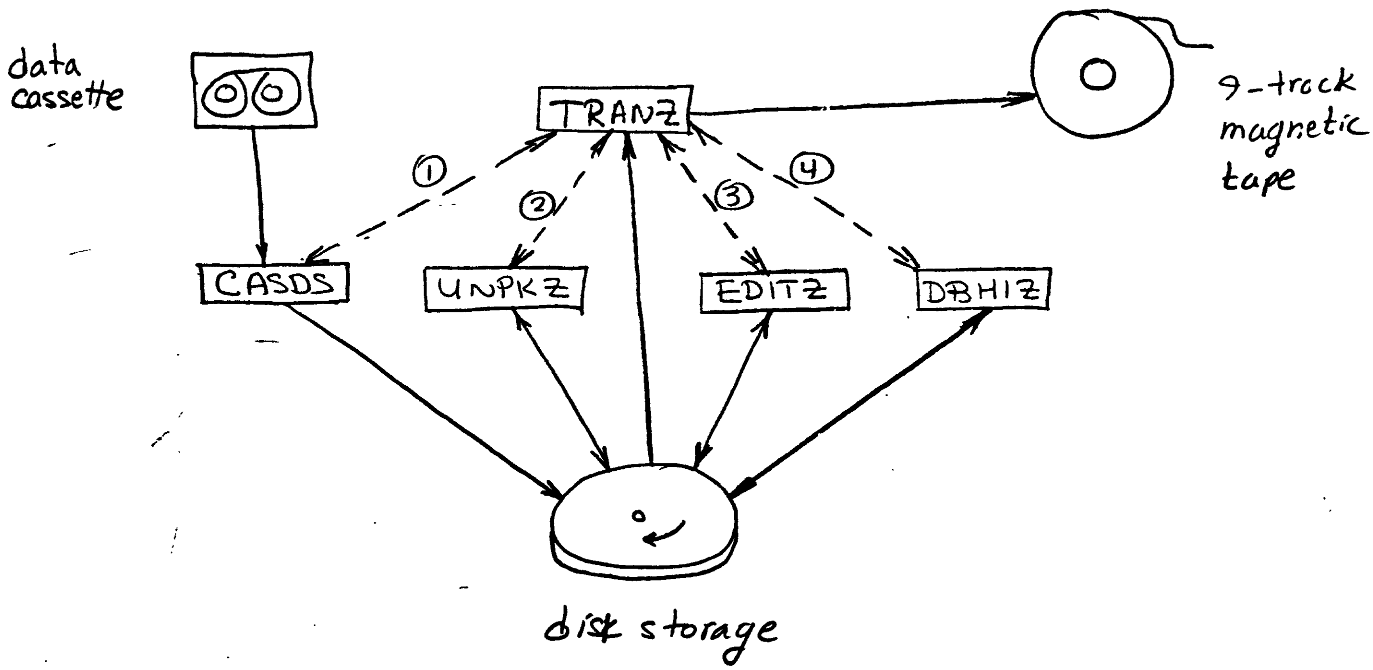
Purpose

Program TRANZ controls the transcription of data recorded on magnetic cassettes by a Sea Data Model 651-2 Data Logger to magnetic tape. The program asks the operator questions about the data to be transcribed and makes certain validity checks on the responses. Most of this information goes into the 128-word header that is written on the source tape. After all questions are answered, control is transferred to program CASDS, which does the actual transfer of the cassette image onto the scratch disk. Upon completion of the transfer, control is returned to TRANZ which sequentially schedules programs UNPKZ, EDITZ, and DBHIZ. These programs unpack, edit, and do a data break and histogram analysis of the data. After these programs have satisfactorily run, TRANZ writes the prepared data onto magnetic tape. The interrelationship of these programs and the data files are shown in Figure 2.1.

Program Description

Program TRANZ is straightforward in nature. The program starts by asking the user to verify that a scratch disk pack is mounted on logical unit (LU) 10. An answer other than "YE" will terminate operation. The save parity error flag, IFLGP, is cleared (IFLGP=0) indicating that records containing parity errors are not to be saved. This flag is set (IFLGP=1) if the user indicates that parity errors are to be saved. The system clock is interrogated to determine the date of transcription. This information along with the version number of TRANZ is reported to the user. Following these housekeeping operations the program enters the main processing loop.

Figure 2.1 Interrelationship of transcription programs. Dashed lines indicate control paths. The circled numbers show the normal sequence of operation. Solid lines indicate the flow of data.



The first task is to clear the 128-word header, IHED, which resides in system common, and set the date of transcription and version number of TRANZ in the header. Next, the tape file number, location code, cassette ID number, instrument number, scan rate, and channels per scan are input by the operator. The operator responses are checked to be certain that they are within allowable bounds. (See Appendix B--User's Guide for a detailed explanation of the input parameters.) If an unallowable response is made, the prompt for the data is repeated.

Standard magnetometer gains are read from a file named MAGAIN which can be created and edited by program MGAIN described in Section 7 of this report. If this file can not be opened, the user is notified and allowed to manually input gain values. If the file can be opened, the gains are displayed. The user is asked if changes are to be made. If changes are made, the new gains will be used in TRANZ, but they will not update the contents of file MAGAIN. When the number of channels exceeds 3, the telluric amplifier gain and line length in meters are also required as input. The final information input is the reset time, off time, stop watch time, and a 50-character comment field.

The program then uses the value for the number of channels (NCHAN) to compute the number of scans per cassette record (NSCAN) and the number of words per unpacked cassette record (NWORD). A summary of this information is given in Appendix A. Several parameters necessary for the transcription are set including the number of subrecords per output record (always 32 and stored in IHED(25)), the number of words per cassette record (IHED(24)), the number of words per output tape record (IHED(26)=IHED(24)*IHED(25)), and the number of characters per cassette record. The last parameter is reported to the user so that the cassette reader can be properly set before the cassette is read.

TRANZ checks the system's Equipment Table Word 5 (EQT5) corresponding to the terminal LU (LUTTY) to be sure the record length message has been printed. When the terminal is available, program CASDS is scheduled and the operator starts the cassette reader. All of the programs scheduled by TRANZ are scheduled "with wait" so that TRANZ will not resume operation until the scheduled program has completed operation. The number of words per tape record (IHED(26)) is passed to CASDS. Upon completion, CASDS returns 5 parameters to TRANZ, which are then reported to the operator. Table 2-1 summarizes the parameters transferred between TRANZ and the programs it schedules. If the completion status (COMST) is equal to 1B, a message saying that both DMA channels were not available will be printed. The call to CASDS will be repeated until both DMA's channels become available. To avoid this situation, it is a good idea not to have other programs running during transcription. A complete diskussion of the completion status word is given in the description of program CASDS.

Each program scheduled by TRANZ returns its version number. All of these version numbers are combined according to the following formula

1000*TRANZ + 100*UNPKZ + 10*EDITZ + DBHIZ

to give a transcription version number. The transcription version number is written in the output header.

Program UNPKZ is used to unpack data from the cassette format. The unpacking process changes the number of words per disk record. Therefore this number, stored in IHED(26), is changed after UNPKZ is finished.

Following the completion of unpacking, the data are edited by program EDITZ. If the editing of the data was satisfactory, the returned value of the completion code flag IFLGC will be 0. Unsatisfactory editing (IFLGC=1) will print a message warning the user that the transcription has been terminated

Table 2.1 Summary of parameters transferred to and from programs scheduled by program TRANZ via calls to system routines RMPAR and PRTN.

Program CASDS

Passed Parameters

1.	IHED (26)	number of words per tape record
2.	not used	
3.	not used	
4.	not used	
5.	not used	

Returned Parameters

1.	CASRC	number of cassette records read
2.	DISRC	number of disk records written
3.	BADRC	number of cassette records with errors
4.	COMST	completion status word
5.	LSTRK	next disk track to be written

Program UNPKZ

Passed Parameters

1.	NDSRC	number of input disk records
2.	NWDSR	number of words per input disk record, IHED(26)
3.	NWCAS	expected number of words per cassette record, IHED(24)
4.	NWSUB	number of words per unpacked subrecord, IHED(52)
5.	not used	

Returned Parameters

1.	JDSRC	number of output disk records
2.	IVER	version number of UNPKZ
3.	not used	
4.	not used	
5.	not used	

Program EDITZ

Passed Parameters

1.	NDSRC	number of input disk records
2.	IFLGP	parity error retention flag
3.	not used	
4.	not used	
5.	not used	

Table 2.1 (Continued)

		Returned Parameters
1.	JDSRC	number of output disk records
2.	IFLGC	completion code flag
3.	IVER	version number of program EDITZ
4.	not used	
5.	not used	

Program DBHIZ

		Passed Parameters
1.	NDSRC	number of input disk records
2.	not used	
3.	not used	
4.	not used	
5.	not used	

		Returned Parameters
1.	NDSRC	number of output disk records
2.	IVER	version number of program DBHIZ
3.	not used	
4.	not used	
5.	not used	

and no data were written onto tape. In this case, TRANZ will jump to the input section and request the next tape file number and prepare for transcribing another cassette.

Satisfactory editing will be followed by the running of program DBHIZ, which performs a data break and clock increment analysis as well as generating a data histogram for use in setting data plotting scales.

When DBHIZ has completed, TRANZ begins writing data to tape. The tape file consists of a 128-word header record followed by data records. (See Appendix A for more information on data formats.) An end-of-file mark is written after the last data record. Following the writing of the output tape, a message is printed on the terminal indicating the transcription version number and the number of data records written. If an end-of-tape mark is encountered while writing the tape, TRANZ will write a message indicating that this has happened and telling the operator to mount a new tape. The old tape is positioned by TRANZ <u>after</u> the previous file mark and a second end-of-file mark is written. The program then pauses while a new tape is mounted. After the new tape is mounted, the program is rescheduled with a *GO,TRANZ command. The processed data are written unto the new tape. After the tape has been written, the program jumps to the input portion to process the next cassette record. If there are no more data to transcribe, a second end-of-file mark is written.

<u>Special Requirements</u>

Before program TRANZ is run, temporary ID segments must be assigned to programs CASDS, ENPKZ, EDITZ, and DBHIZ to prevent SC05 scheduling errors from occurring. This is done with the FMGR Restore Program (:RP) command which is invoked by transfer file /TRANZ (See Section 8).

The system clock should be set to the correct day and year before transcription is begun since this information is placed in the tape header record. Refer to the HP RTE/II Software System Programming and Operating Manual for details. An example is also given in Appendix B.

A scratch disk cartridge must be loaded on LU 10, but not mounted with the FMGR Mount Cartridge (:MC) command. Issue a FMGR Cartridge List (:CL) command to see if LU 10 is mounted. If it is mounted, use the Dismount Cartridge (:DC,-10) command to dismount the cartridge before running TRANZ. If the cartridge you have just dismounted is not a scratch pack, replace it with a scratch pack as any data on it will be destroyed when TRANZ is run.

The track and sector addresses used by TRANZ for reading data from disk are based on 96 sectors per track. If the disk drive used has a different number of sectors per track, an appropriate change will have to be made in the address incrementing code.

Program Loading

Program TRANZ requires no additional relocatable modules at load time. It must, however, be loaded with system common using the commands:

```
:LG,2
:MR,%TRANZ
:RU,LOADR,99,6,10,0,2
:SP,TRANZ
```

The operating system must have been generated with a least 128 words of background system common.

Program Operation

The program is run with the command

```
:RU,TRANZ
```

The program will ask the user for various pieces of input data. See Appendix B--User's Manual for details.

3. CASDS - Cassette to Disc Transfer Program

Purpose

Program CASDS is used to transfer data from the Sea Data Model 12 cassette reader to the HP7900A disk drive. The reader usually outputs a data word every 500 microseconds, with some data coming as often as 175 microseconds. The HP RTE-II operating system cannot handle interrupts at this high rate without special hardware. To achieve this high data-transcription rate it was necessary to turn off the interrupt system and use asynchronous DMA input and output from separate buffers.

Program Description

The program uses two buffer areas BUFA and BUFB, which are asynchronously filled and emptied via DMA control. When BUFA is full, the second buffer BUFB starts filling. Simultaneously, BUFA is being written to disk. After BUFA is empty, the program waits for BUFB to fill at which time BUFA starts filling while BUFB is written to disk. This loop of events continues until all the data have been transcribed.

While the program is waiting for a buffer to fill, it checks the incoming data for reader errors, end-of-record marks, and the operator stop signal. A tally of the number of bad records encountered is kept. If an operator stop condition is encountered, the active input DMA is halted. The unfilled portion of the current input buffer is written over with zeros, and the entire buffer is written to disk. Writes to the disk are always 1024 words long even though the amount of the buffer actually used is only WDDS words long.

The operator signals the program when the data transcription is to stop by setting bit 15 of the CPU Display Register. CASDS then returns five parameters to program TRANZ: 1.) CASRC - the number of cassette records read, 2.) DISRC - the number of disk records written, 3.) BADRC -

the number of cassette records marked by the reader as bad because of either a parity, short record, excess data, or low signal error (see Appendix A for more information), 4.) COMST - the completion status word (diskussed below), and 5.) TRACK - the track address the next disk write would have occurred on. For a normal completion with no problems, the completion status word COMST will be 0B. If both DMA channels are not available, the program will terminate with COMST=1B, no data will have been transferred, and TRANZ will then try to reschedule CASDS. COMST will be set to 177777B if the disk has been completely filled. In the event of a disk error both COMST and the S-register will be set to the disk status word. The CPU will halt (HLT 77B). The meaning of the disk status word can be found in the HP disk interface manual.

Special Requirements

The select code of the disk drive and cassette reader is stored in locations DC, CC, and CASS respectively. These locations are located in the DEVICES, CONTROL WORDS, AND COMMAND section of the program. The last two octal digits of control word CW1IN should be set to the cassette reader select code. Similarly, the last two octal digits of control word CW1OT should be set to the select code of the disk drive data channel. These locations should be modified to correspond to the input/output configuration of the system the software will be run on before CASDS is assembled.

There is a provision between labels CHECK and STEOF for automatic detection of end-of-file mark during cassette reading. If the proper hardware is installed in the cassette reader, which sets bit 14 of the reader Message Word when an EOF is encountered, the asterisk at the beginning of line 245 can be removed. This will allow the program to automatically terminate transcription without operator intervention when the end of a cassette has been reached.

Program Loading

No additional relocatable modules are used by CASDS. The loading procedure is as follows:

```
:LG,2
:MR,%CASDS
:RU,LOADR,99,6,0,0,2
:SP,CASDS
```

Program Operation

In normal operation CASDS is scheduled by program TRANZ which passes parameter WDDS to CASDS. WDDS is the number of reader words to be stored in each 1024-word-long disk record. After some housekeeping functions have been performed, transfer of data to the internal buffers begins. The operator will notice that the INTERRUPT SYSTEM has been turned off, and that the S-register display is flashing as data is being read. The S-register displays information concerning where the next record will be written on the disk. Bits 11-3 are the octal track address (OB-310B); bit 2 is the head address (0 for the upper surface, and 1 for the lower surface of the scratch disk); and bits 1 and 0 are the sector address (OB = sector 0, 1B = sector 8, and 2B = sector 16. One sector is 128 words long). The display will go through six different sector-head combinations before the track address is incremented.

The operator can signal to the program via the S-register when the end of the cassette has been reached. Setting bit 15 of the display will cause CASDS to halt and control to transfer back to TRANZ. This might be done if, for example, the wrong cassette was being transcribed, or if only part of the cassette needed to be transcribed. When the cassette has been fully transcribed as indicated by the PERCENT of REFERENCE meter level dropping to zero, bit 15 should be set by the operator. This will cause CASDS to terminate. The STOP and REWIND buttons on the cassette reader can now be pressed. After the cassette has rewound, it can be removed from the reader.

4. UNPKZ - Data Unpacking Program

Purpose

Program UNPKZ unpacks the Sea Data cassette image now on disk from the cassette reader format to source tape format. The input data are searched for end-of-record (EOR) marks, checked for proper length, unpacked, and then written to disk. Special action is taken whenever the record length is not correct. A summary of the actions taken during processing is printed.

Program Description

UNPKZ is passed several parameters by program TRANZ: 1.) NDSRC - the number of input disk records, 2.) NWDSR - the number of words per input disk record, 3.) NWCAS - the expected number of words per cassette record, and 4.) NWSUB - the number of words per unpacked subrecord. Figure 4.1 contains a flow diagram of UNPKZ, which may be helpful in understanding the following discussion.

The program enters the main processing loop and increments the input track (ITRK), input sector (ISEC), and input disk record (IDSRC) pointers. A disk record is read into the input buffer IBUF behind any previous data. The location of the last data word (LEFT) is incremented and the input buffer pointer (IPT) set to zero.

Routine FIND is now used to search for an EOR mark which is designated by a word with bits 15 and 0 set. The search goes from location IPT+1 to LEFT+1 of array IBUF. Upon exit, IEXIT is set to the number of words found in the record, or -1 if no EOR was encountered. If the record contains the proper number of words (IEXIT=NWCAS) the record is unpacked using routine UNPAK and placed in output buffer JBUF. The seventh word of the output

Figure 4.1 Flow diagram of program UNPKZ.

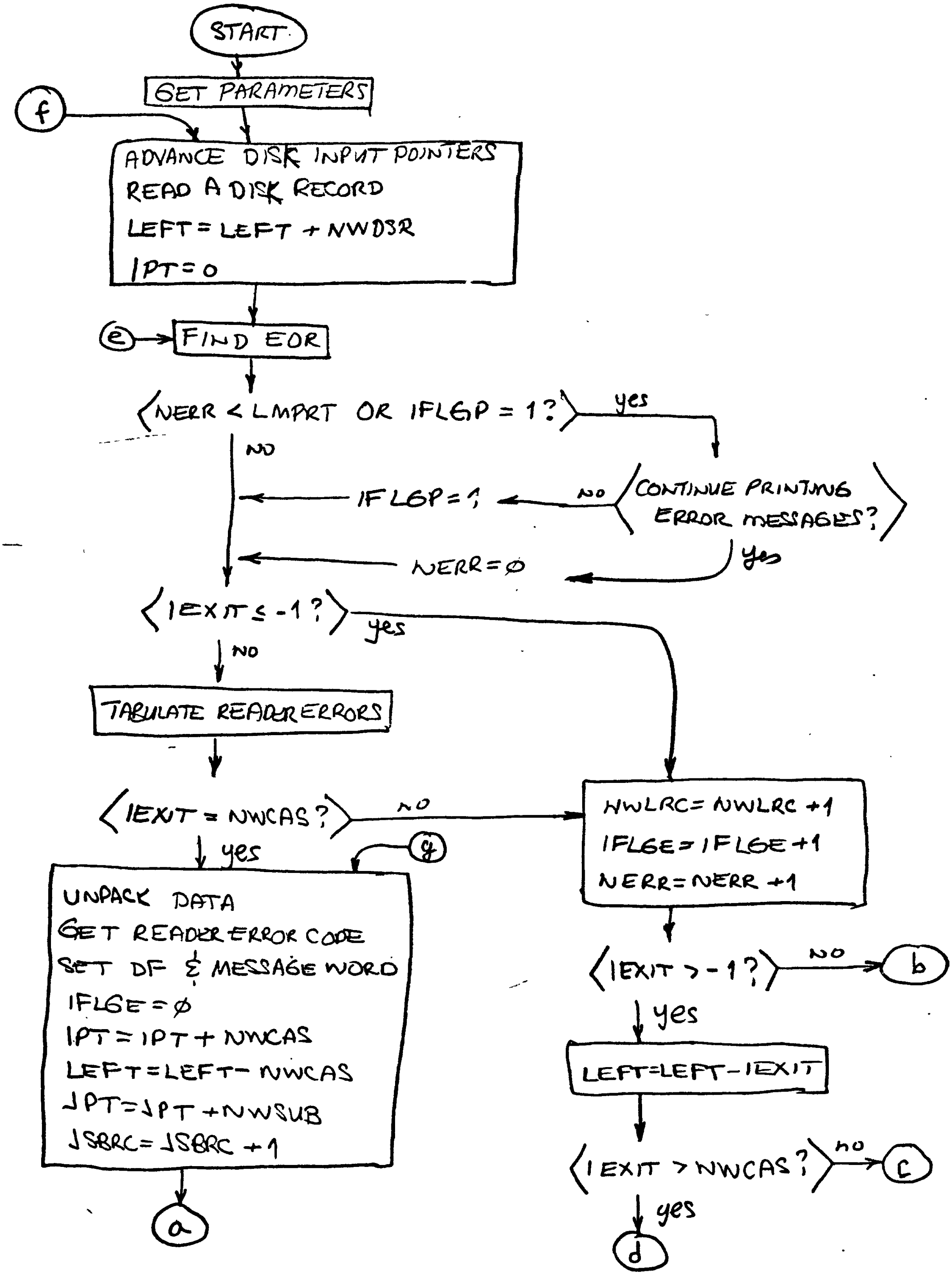
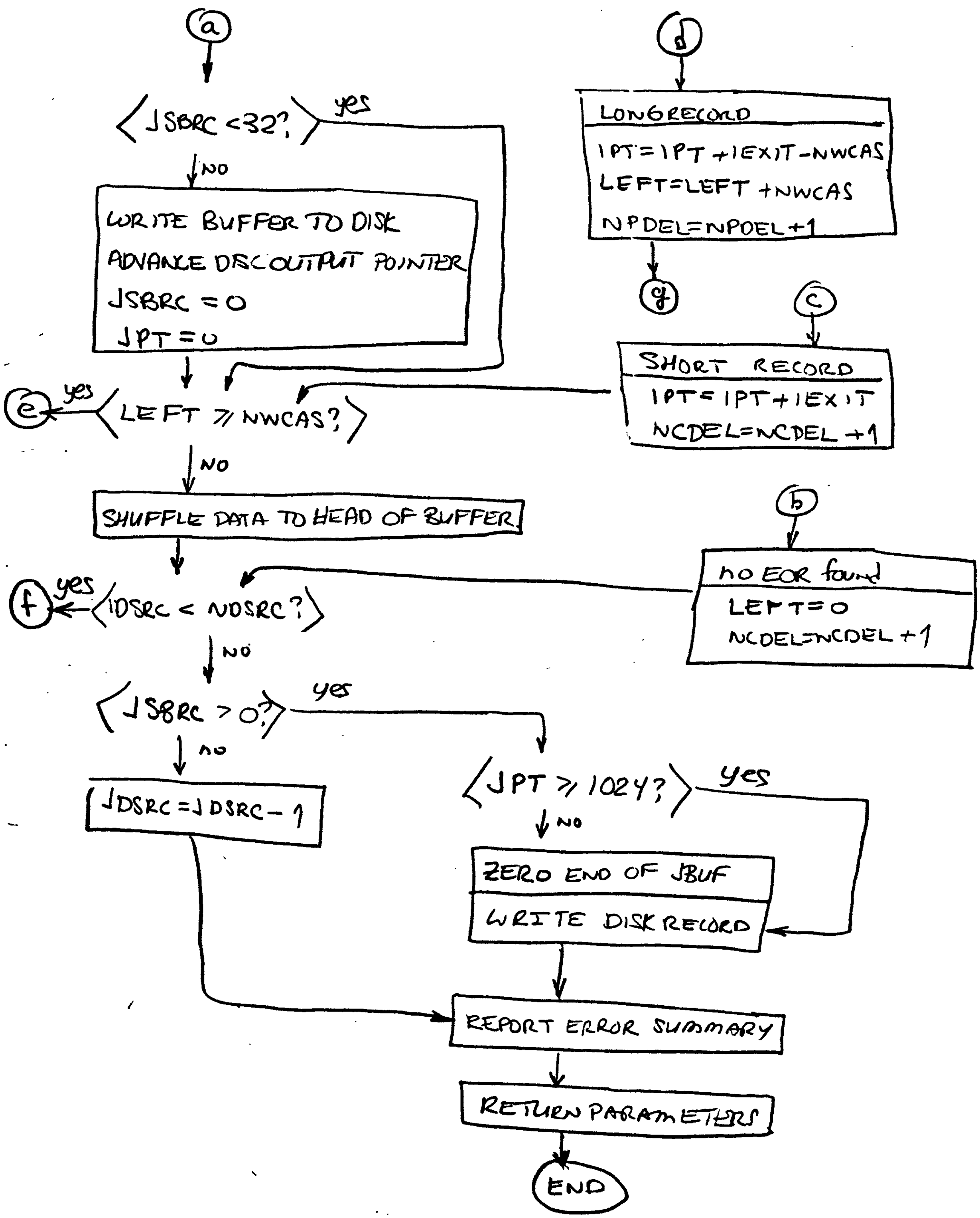


Figure 4.1 Continued



subrecord is set as follows:

bits 15-9 = minimum number of cassette records since previous subrecord, IFLGE

bits 8-5 = reader message word

bits 4-1 = reader parity word

bit 0 = data flag

A complete description of these bits is given below. The error flag (IFLGE) is cleared, the input and output pointers (IPT, JPT, and JSBRC) are advanced, and the remaining word count (LEFT) is decreased.

If the output buffer is full, it is written back to disk and the output pointers zeroed. If the input buffer contains less than NWCAS words, the remaining words are moved to the head of the buffer. If there are more data on disk, the program goes to the beginning of the main processing loop. If there are no more unpacked data on the disk, the end of the output buffer is zeroed, and the entire buffer is written to disk. The program prints a summary containing the following: 1.) the number of input disk records (NDSRC); 2.) the number of output disk records (JDSRC); 3.) the number of wrong-length cassette records (NWLRC); 4.) the number of complete record deletions (NCDEL); 5.) the number of partical record deletions (NPDEL); 6.) wrong-length cassette records (NWLRC); 7.) the number of parity, short, low signal, and excess data errors encountered. These errors are tabulated <u>only</u> for cassette records that did <u>not</u> contain the proper number of words.

When a cassette record is encountered which does not contain the proper number of words, a different processing procedure is used. First the wrong length record count (NWLRC) and the error flag (IFLGE) are incremented. If IEXIT is equal to or less than -1, meaning no EOR was found, the data are deleted, LEFT is set to zero, the complete deletion counter (NCDEL) is incremented, and the program begins processing the next input disk record. If

IEXIT is positive, any reader errors are tabulated and LEFT is reduced by IEXIT. If IEXIT is less than NWCAS, the entire cassette record is deleted, and the complete deletion counter (NCDEL) is incremented. A "SHORT RECORD - DELETE" message is generated. However, if IEXIT is greater than or equal to NWCAS, the difference IEXIT - NWCAS is deleted and the last NWCAS words are processed as a normal record. The partial deletion counter (NPDEL) is incremented. A "LONG RECORD - PARTIAL DELETE" message is printed.

All error messages contain the input disk record number (IDSRC), the input buffer pointer (IPT) of the offending record, and the expected number of words per cassette record (NWCAS). Some messages will have the number of words found in a cassette record (IEXIT), the number of words left in the input buffer (LEFT), or the cassette reader status word (PSLE). PSLE will be a 4-digit number with 1's corresponding to the type or reader error (PE, SH, LO, and EX) encountered.

Only the first LMPRT (100) error messages are printed. At that time the user is asked if further error messages are desired. The user can respond "YE" or "NO" to continue or terminate the messages respectively. If the message are continued, the user will be again asked after another LMPRT messages have been printed, if the messages are to continue.

The seventh word of the output subrecord warrants further diskussion. The format of this word was described above. The data flag bit (bit 0) is set or cleared by hardware in the Sea Data 651-2 data logger. It is presently not used. Bits 8 through 5 are the reader message bits P, S, L, and E. Bits 4 through 1 are the reader parity bits. (See Appendix A for a complete description.) Bits 15 through 9 are the value of IFLGE for the subrecord. IFLGE corresponds to the number of cassette records dropped since the previous subrecord. They are also referred to as missing records in program EDITZ.

The hoped-for value of IFLGE is zero, but data errors will cause this number to increase. There is some ambiguity in the interpretation of this number since it can be <u>smaller</u> than the number of cassette records deleted. If, for example, NWCAS is equal to 30, and 63 words are found in the record (IEXIT=63), UNPKZ will throw away the first 33 words and unpack the last 30 words. The ambiguity comes about because there is no way of telling if the 33 words which are deleted correspond to 1, 2, or more cassette records. Thus IFLGE always gives the minimum number of missing subrecords. This statement is based on the assumption that the cassette reader does not generate extraneous EOR marks, and that the cassette recorder has not written extraneous EOR marks on the cassette. Either of these situations will cause IFLGE to be too large.

Special Requirements

Disc reads and writes are accomplished by EXEC calls. Track and sector addresses are computed by routine NXTRC, which assumes that here are 96 sectors per disk track. This corresponds to an HP7900A disk drive. If a different disk drive is used, an appropriate change will have to be made in subroutine NXTRC.

Assembly language routines FIND, UNPAK, and MOVE use instructions which are only found on HP-21MX and new CPU's. These instructions will have to be simulated if the programs are run on an older processor.

Program Loading

Program UNPKZ uses assembly language routines FIND, UNPAK and MOVE which must be provided at load time. The loading procedure is:

```
:LG,2
:MR,%UNPKZ
:MR,%FIND
:MR,%MOVE
```

```
:RU,LOADR,99,6,0,0,2
:SP,UNPKZ
```

Program Operation

Program UNPKZ is scheduled by TRANZ. It requires no operator action. A detailed description of the output is provided in Appendix B--User's Manual.

5. EDITZ - Data Editing Program

Purpose

Program EDITZ is used to edit the unpacked geomagnetic subrecord headers. The headers are scanned for the occurrence of the following errors: 1.) parity error (PR), 2.) inconsistent header (IC), 3.) short record (SH), 4.) low reader signal level (LO), 5.) excess data (EX), and 6.) improper clock incrementation (CI). The program reports and tries to correct most errors. Operator intervention is requested for the correction of certain clock incrementation errors that the program is not "smart" enough to handle. The program does not make any changes to the data portion of the subrecords.

Program Description

EDITZ starts by receiving the following parameters from program TRANZ: 1.) NDSRC - the number of input disk records, and 2.) IFLGP - the parity error retention flag. The program then computes the proper clock increment per subrecord (DT=NSCAN*2**NRATE). The output clock pointer JPTC is set to the value of NEXT. NEXT, which is set in a data statement, must be greater than 1 and less than 9 for proper program operation. NEXT controls the amount of data output when clock incrementation errors occur.

Before the processing loop is diskussed, it would be appropriate to describe the data storage registers used by EDITZ (see Figure 5.1). Input data from the disk is read into buffer IBUF. The i-th word of the current subrecord is stored at location IBUF(IPTD+i). Processed subrecords are stored in JBUF with the j-th location of the next subrecord to be output stored at JBUF(JPTD+j). The number of input subrecords left to be processed is call LEFT. Array IMISR contains the number of missing or removed cassette records since the previous cassette record corresponding to the data presently in IBUF. More specifically IMISR(IPTC+1) contains bits 14-9 of IBUF(IPTD+7).

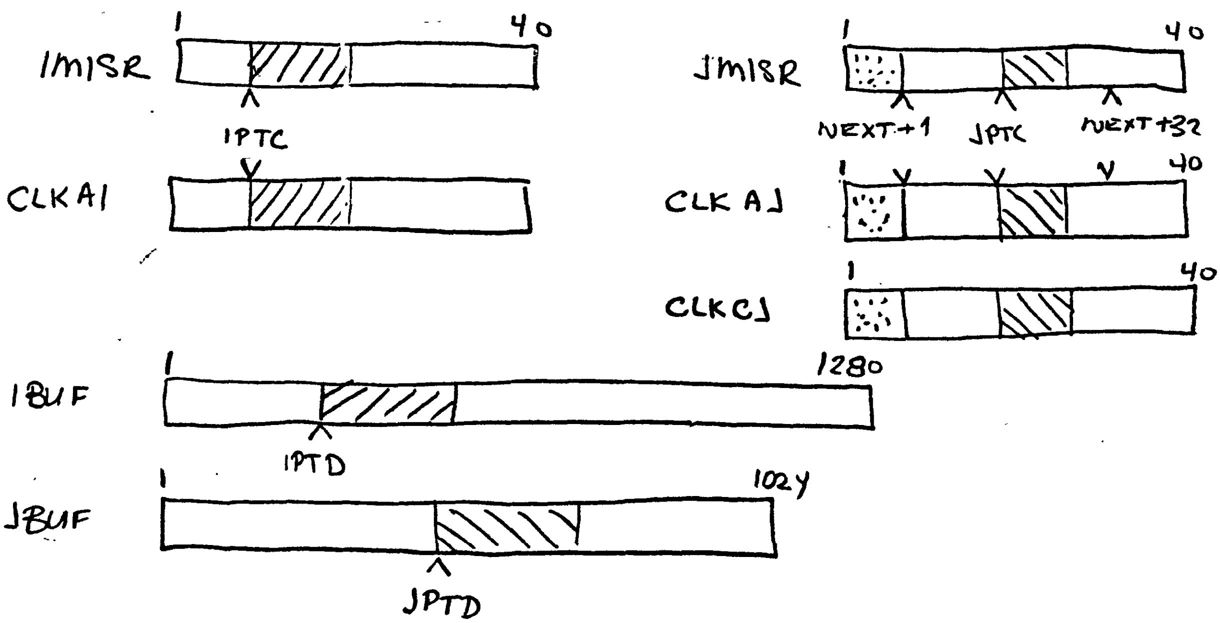
Figure 5.1 Storage arrays used by EDITZ.

The current input subrecord being processed begins at IBUF(IPTD+1). The corresponding values of the number of missing records and actual clock values are stored at IMISR(IPTC+1) and CLKAI(IPTC+1) respectively. The next missing record count, actual clock, and corrected clock will be output at JMISR(JPTC+1), CLKAJ(JPTC+1), and CLKCJ(JPTC+1), respectively.

data currently being processed

output area for next successfully processed record

processed data previously written to disk



pointer	initial value	increment
IPTC	0	1
IPTD	0	NWORD
JPTC	NEXT	1
JPTD	0	NWORD

The value of IPTD is NWORD times IPTC. Array CLKAI contains the clock values corresponding to the data in IBUF i.e., CLKAI(IPTC+1)=32768*IBUF(IPTD+1)+IBUF (IPTD+2).

As data are processed, they are moved to buffer JBUF. The next output subrecord will start at location JBUF(JPTD+1). The initial value of JPTD is zero. Location JMISR(JPTC+1) contains the number of missing records corresponding to the value of bits 14-9 of word JBUF(JPTD+7). Array CLCKJ contains the corrected clock values of subrecords which have already been processed. These values correspond to the corrected clock values in JBUF; more specifically, CLKCJ(JPTC+1)=32768*JBUF(JPTD+1)+JBUF(JPDT+2). Array CLKAJ contains the actual clock values before they were corrected. Sometimes the values in CLKAJ and CLKCJ are the same. The output pointer JPTC is initially set to NEXT. This is to allow the first NEXT locations of JMISR, CLKAJ, and CLKCJ to be used for previously processed data. This data is displayed by routine UHELP whenever user input of clock values is requested.

When the amount of unprocessed input data becomes too low (LEFT less than NEXT), the unprocessed words in IMISR, CLKAI, and IBUF are moved to the head of their respective arrays. For the first two arrays this amounts to LEFT words, while for IBUF this is NWORD*LEFT words. A new disk record is read into IBUF beginning at location IBUF(NWORD*LEFT+1) and unloaded beginning at IMISR(LEFT+1) and CLKAI(LEFT+1). The input pointers IPTC and IPTD are reset to zero.

The program begins the general processing loop by reading and unloading a disk record (see Figure 5.2). If IFLGD is set, the missing record count is incremented and IFLGD is cleared. If all the data have been processed, the

Figure 5.2 Flow diagram of program EDITZ.

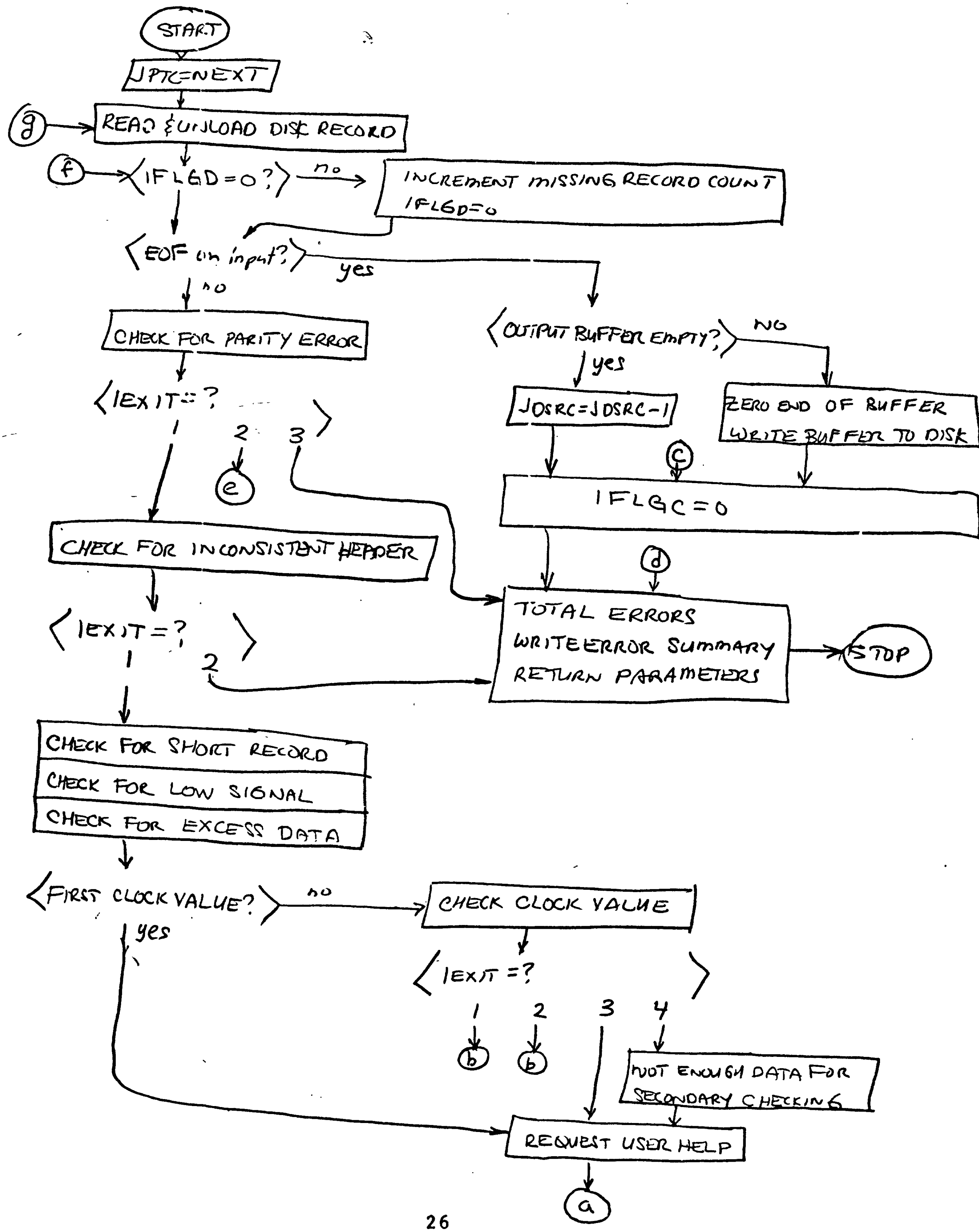
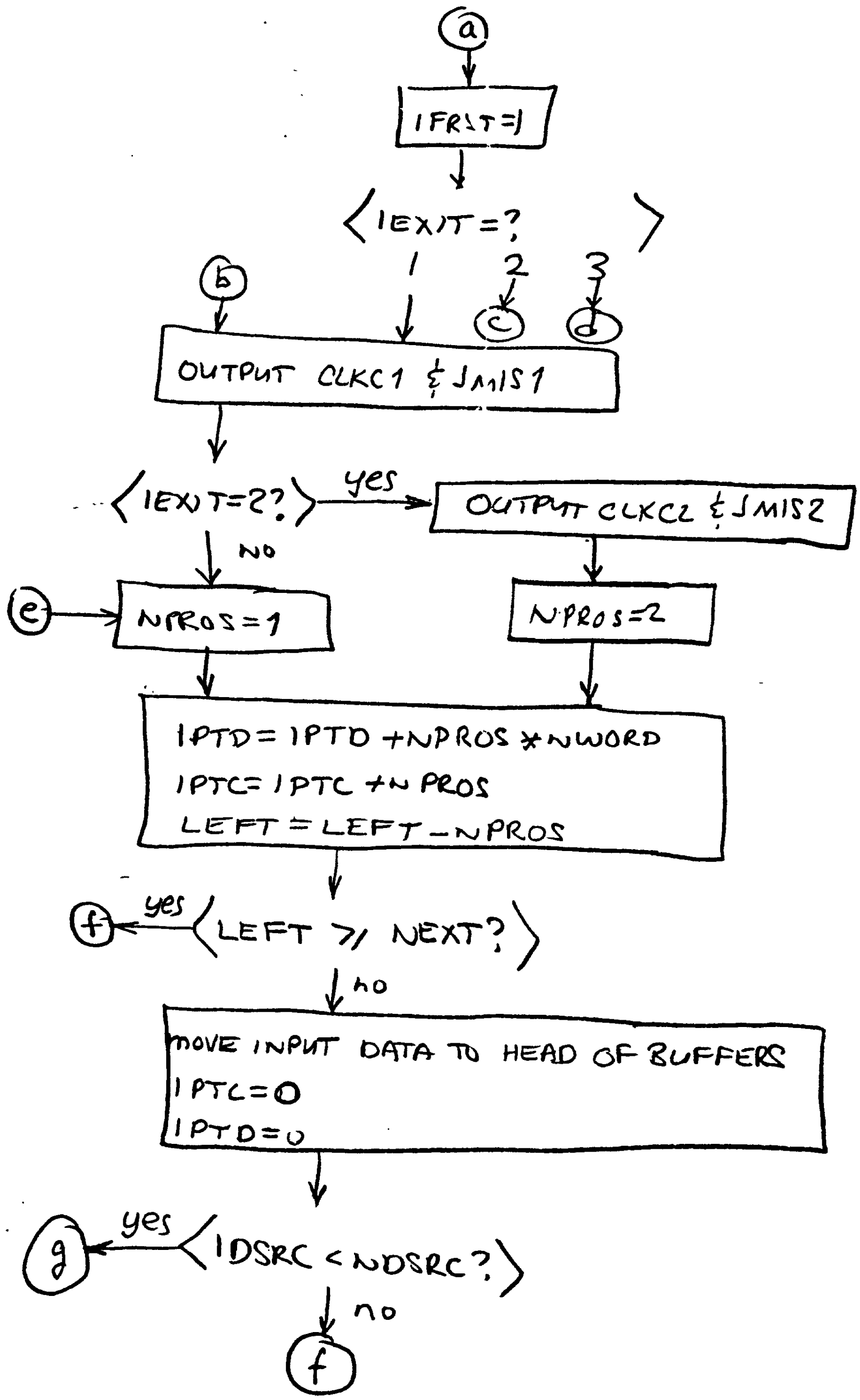


Figure 5.2 Continued



program flushes the output buffer, reports the error statistics, and returns the number of output disk records (JDSRC), the completion code flag (IFLGC) and the version number before terminating.

If all of the data have not been processed, checks are made for parity (PR) errors, inconsistent headers (IC), short record (SH) errors, low signal (LO) errors, excess data (EX) errors, and improper clock incrementation (CI). The 4 errors (PR, SH, LO, and EX) are detected by looking at the cassette reader message word bits 10-7 which have been placed in bits 8-5 of the 7th subheader word (IBUF(IPTD+7) by program UNPKZ. Separate subroutines are used to detect and handle the various error conditions. These routines are described below.

Parity errors are handled by subroutine PARIT. There are two basic modes of operation. If IFLGP is set, subrecords with parity errors are eliminated, while if IFLGP is clear the data are retained. No attempt is made to try and correct the parity error. A message is printed whenever a parity error is detected. After the first LPRNT messages (set to 100), no further messages are printed. The parity error message contains the input disk record number (IDSRC), the value of IPTC, and value of CLKAI, and the cassette reader parity check word (PWORD). PWORD is displayed as a 4 character octal number, the characters referring to the 4 cassette data tracks. A character is clear (zero) when the parity error occurred on the corresponding track.

When data with parity errors are being saved, a differential error count (IERP) is kept. When IERP exceeds LMPR (the parity error limit which is set in EDITZ to 10, the user is asked if the processing should continue. A response of "NO" sets IEXIT to 3, and processing of the present cassette is terminated. If the user responds with "YE" the differential error count is zeroed, and LMPAR is doubled. Upon exit from PARIT, IEXIT equals 1 for data retention, 2 for data elimination, and 3 for processing termination.

Subroutine INCON checks header consistency and makes the necessary corrections. An inconsistent header condition exists whenever words 3, 4, 5, or 6 of the subrecord header do not equal words 7, 8, 9, and 10 respectively of the header stored in system common (IHED). These words represent the values of 1.) cassette ID; 2.) instrument number, 3.) scan rate, NRATE; and 4.) number of channels per scan, NCHAN. The user furnishes the values in IHED as input to program TRANZ.

When an inconsistent header is detected, the values stored in IHED are used to replace the subrecord header values. An error message is generated for the first LMIC occurrences. LMIC is set in EDITZ to 5. When the number of IC errors exceeds LMIC, the user is shown the latest values of the subheader and the values in IHED, and asked if processing should continue. At this point the user should check the values input to TRANZ to see if any typographical errors were made. The user is also asked if processing should continue. If the user responds "NO", IEXIT is set to 2 and processing of the current cassette image is terminated. A response of "YE" will continue processing, but no more IC errors will be reported. In this and all other cases, IEXIT will be set to 1 on exit.

Short record errors (SH) are noted by subroutine SHORT. If no error condition exists, IEXIT is set to 1. When an error is detected, it is noted, and IEXIT is set to 2. The present version of EDITZ treats both of these conditions the same; that is, the data are retained. This is done because program UNPKZ retains only records which are of the proper length without looking at the value of the S bit in the reader message word.

Low signal errors (LO) are reported by subroutine LOSIG. The first LMLO (set to 10) errors of this type are reported. Similar handling of excess data errors (EX) is provided by subroutine EXCES. The only difference is that the reporting limit is called LMEX and it is set to 5. The low signal and excess data error conditions are not considered to be very reliable; therefore, the data are not eliminated.

Clock incrementation is checked by subroutine CHECK. This probably is the most complicated and critical test performed on the data. The clock should increment each subrecord by an amount DT which equals NSCAN*2**NRATE. Subroutine DIFF is used to compute the clock difference (DIFF1) between the current and previous clock value, the integral number of multiples of DT contained in DIFF1 (I1), and to set flag INFL1 if DIFF1 equals I1*DT (See Figure 5.3). The difference DIFF1 is computed using function DIF20 which adds 1,048,576 (2**20) to the difference if it is less than -1,000,000. This is done to allow for roll-overs in the 20-bit recorder clock, but not allow clock decrements. Parameters IMSR1 and IMSR2 are the estimated number of missing records between the current and previous clock values, and the current and next clock values, respectively.

Subroutine CHECK has several exit conditions indicated by the value of IEXIT that corresponds to the severity of the condition.

The list below contains the test applied to the data, and the value of the corrected clock (CLKC1) and number of missing records (JMIS1) used if the test is passed:

1. The clock increment is checked for the expected value of DT.

   Test: I1=1 and INFL1=1?

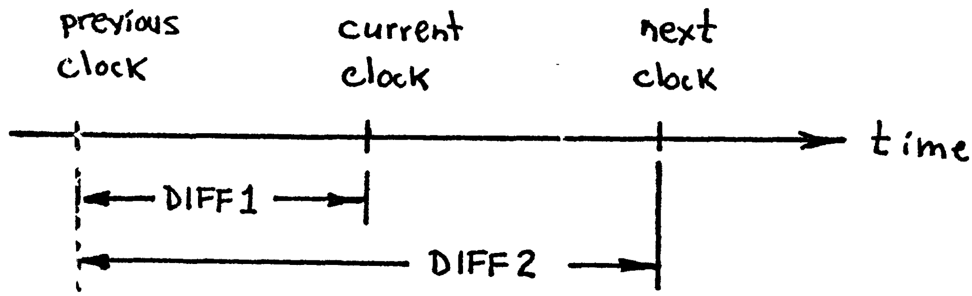
   Action: JMIS1=0

   CLCK1=CLKCJ+DT

   Most data should pass this test.

If this test is passed, IEXIT is set to 1, CHECK is exited, and the data are output.

Figure 5.3 Clock values and differences used by subroutine CHECK.



The next set of tests rely upon secondary clock differences, that is, the difference between the previous and next clock value. The variables DIFF2, I2, and INFL2 refer to the same quantities as described above, but for the secondary clock difference. Before the secondary difference variables are formed, the program checks to be certain LEFT is greater than or equal to 2. If there are not enough data left to perform the test, IEXIT is set to 4. The main program will then print a message indicating there is not enough data for a secondary clock difference test, and ask for user input.

If the secondary difference can be formed, the following tests are performed:

2. Check for the secondary difference being a multiple of DT, and greater than or equal to 2*DT.
   Test: INFL2=1 and I2≥2?
   Action: If this test fails, set IEXIT to 3 and exit. User intervention required.
3. Check for more than 4 missing cassette records. Don't allow the program to make corrections for too large a data break.
   Test: I2>6?
   Action: If this test fails, set IEXIT to 3 and exit. User intervention required.
4. Check if the middle clock position can be resolved by IMSR1.
   Test: IMSR1≤I2-2?
   Action: JMIS1=IMSR1
   CLKC1=CLKCJ+DT*(JMIS1+1)
   JMIS2=I2-JMIS1-2
   IEXIT=2
5. Check if the middle clock position can be resolved by IMSR2.
   Test: IMSR2≤I2-2?
   Action: JMIS2=IMSR2
   JMIS1=I2-JMIS2-2
   CLKC1=CLKCJ+DT*(JMIS1+1)
   IEXIT=2

6. Since all of the previous tests have failed, arbitrarily set JMIS1 to zero and adjust JMIS2 and CLKC1 accordingly.

Action: JMIS1=0

JMIS2=I2-2

CKC1=CLKCJ+DT

IEXIT=2

An exit condition (IEXIT) of 1 indicates that first clock differences were used, 2 indicates secondary clock differences were used, 3 indicates the program could not resolve the clock ambiguity, and 4 means that there was not enough data to do secondary clock differences.

Clock incrementation errors (IEXIT equal to 3 or 4) require user assistance to correct the clock values. This is provided by routine UHELP which displays the previous NEXT actual (CLKA) and corrected (CLKC) clock values, and the number of missing records (MISR). Also displayed are the NEXT future values of the clock and missing record values. Differences of the clock are also displayed to help the user in determining the best value to set the clock to. UHELP computes a best clock estimate for the next clock value, which is the previous corrected clock value plus DT times one more than the number of missing records. The user is asked if the best clock estimate should be used. The clock will be changed to the best estimate by responding "YE". A response of "NO" will require the user to input a clock value. On all but the first clock value processed, UHELP determines if the value supplied by the user differs from the previous clock value by an integral multiple of DT which is greater than or equal to 1. If it does not, the user will be asked to input another clock value. The first clock value is not subjected to this test. This allows the user to introduce a constant positive or negative shift to all clock values. This might be done to correct for known errors in the clock reset times. After UHELP has accepted the clock value, it uses it to adjust the number of missing records to correspond to the clock differences.

A response of "ST" (for STOP) to the question of using the best clock estimate will halt processing and set IEXIT to 2. The user is then asked if the processed data should be saved. A response of "NO" sets IEXIT to 3 and the data will be lost. Any response other than "NO" will set IEXIT to 2 which saves the processed data.

Special Requirements

The track and sector addresses are computed by routine NXTRC, which assumes an HP7900A disk drive with 96 sectors per track is being used. If another disk drive is used, an appropriate change will have to be made to this routine.

Routine MOVE makes use of HP-21MX instructions. Use of an older CPU will require the simulation of these instructions.

Program Loading

EDITZ uses subroutines PARWD and MOVE, which must be supplied at load time. The program must be loaded with system common. The suggested load procedure is:

```
:LG,2
:MR,%EDITZ
:MR,%PARWD
:MR,%MOVE
:RU,LOADR,99,6,10,0,2
:SP,EDITZ
```

Program Operation

Program EDITZ is scheduled by TRANZ. The operator is requested to make clock corrections that the program is not able to fix, and make decisions about whether or not the processing should be continued. See Appendix B—User's Manual for examples of program operation.

6. DBHIZ - Data Break/Histogram Analysis Program

Purpose

Program DBHIZ is used to perform a data break and histogram analysis of the edited data on disk before it is written to magnetic tape by program TRANZ. The data break analysis reports irregularities in the clock values. This is done as a check on the data editing of program EDITZ, and to give the user an idea of the quality of the data. The data histogram analysis is used to obtain guidelines for setting scales for data plotting routines. The histogram analysis also provides a means of quickly determining if major problems, such as an off-scale magnetometer, exist in the field instruments.

Program Description

DBHIZ receives one parameter from TRANZ: the number of input disk records (NDSRC). The program computes gain factors for each data channel, which are used to convert from data counts to data values via the formula:

$$V = G*(C - 2048)$$

where V is the data value, G the gain, and C the number of counts. For magnetometer channels 1, 2, and 3, the gain is given by

$$G(I) = IHED(I+53)/2048 \text{ nT/count, } I=1,3.$$

If IHED(I+53) equals zero, the gain defaults to a value of 0.4882813 nT/count. The telluric channel gains are given by

$$G(I)=4882.813/IHED(I+56)/IHED(I+58) \text{ mV/km/count, } I=4,5.$$

In the event any of the denominator terms are zero, the gain is set equal to 0.0 mV/km/count. The gains for channels 6 and 7, which are normally not used, are set to 0.4882813 units/count. The program also computes the expected clock increment per subrecord, DT. This equals the number of scans per subrecord times the scan interval in ticks (2 ticks = 1 second).

The program reads through the data record by record. The incremental clock change is checked for every subrecord. If the incremental clock change is equal to DT and there is no data break (indicated by the fourth word of the subrecord being non-negative), no error message is printed. Clock increments less than or equal to -1,000,000 have the value 1,048,576 added to them to accomodate rollovers in the 20-bit data-logger clock. Whenever the clock increment does not pass this test, the following information is printed:

1. IDSRC - disk record number
2. ISRC - subrecord number
3. CLKI - previous subrecord clock value
4. CLKJ - present subrecord clock value
5. DIFF - difference of CLKJ and CLKI modulo 1,048,576
6. MISS - number of missing subrecords
7. IDB - data break indicator (equals "DB" for data break)

On the same pass through the data, a histogram analysis is also carried out. The data for each channel is tabulated in one of 32 bins which are each 128 counts wide. The first bin begins at 0 counts, and the last bin ends at 4095 counts. The final output includes the bin number, the bin bounds in counts and channel units (nT or mV/km), the number of data points in the bin, and percentage of data in the bin.

When DBHIZ has terminated it returns the number of disk records processed (NDSRC) and its version number. A completion message is also printed indicating the value of NDSRC and the number of data breaks (NDB).

Special Requirements

The track and sector addresses are computed by routine NXTRC, which assumes an HP7900A disk drive with 96 sectors per track is being used. If another disk drive is used, an appropriate change will have to be made to this routine.

Program Loading

DBHIZ requires no other modules at load time. However, it must be loaded with system common since the programs needs access to the header IHED. The loading procedure is:

```
:LG,2
:MR,%DBHIZ
:RU,LOADR,99,6,10,0,2
:SP,DBHIZ
```

Program Operation

Program DBHIZ is scheduled by TRANZ. It does not require operator action. A detailed description of the output is provided in Appendix B - User's Manual.

7. MGAIN - Magnetometer Gain File Editor

Purpose

Program MGAIN is used to create and edit the magnetometer gain file. The magnetometer gains are used to convert data-logger counts to nanoteslas (gammas).

Program Description

The magnetometer gain file, MAGAIN, can be created and edited by program MGAIN. When run, the program tries to open a file called MGAIN with a security code of 518. If the file can not be opened, it is created on logical unit 2, the system disk. The file is 1 block long (128 words) with records 3 words long. The file is a File Manager type 2 file. The first record contains the day, month, and year the file was last changed, followed by 31 gain records. Each gain record contains the gains for the X, Y, and Z channels. The range of the magnetometer in nanoteslas is between the gain value and its negative. After the file is created, the date is furnished by the operator, along with the gains for all the instruments. Following the last gain input, a listing of the gain file is printed.

If the gain file already exists, the program prints the date of the last update, and then allows changes to be made to the file. If changes are made, the current date is requested. A listing of the gain file is then printed.

Special Requirements

The magnetometer gain file is named MAGAIN. If another program uses a file with this name there is a potential conflict. The user should avoid creating a file on another disk cartridge with the same name.

Program Loading

MGAIN requires no other load modules. The loading procedure is:

```
:LG,2
:MR,%MGAIN
:RU,LOADR,99,6,0,0,2
:SP,MGAIN
```

8. Transfer Files /TRANZ and \TRANZ

Purpose

The programs scheduled by program TRANZ must have temporary ID segments assigned to them before TRANZ is run to prevent the occurrence of SC05 scheduling errors. Transfer file /TRANZ is used to restore the appropriate programs; that is, assign temporary ID segments. When transcription has been completed, transfer file \TRANZ is used to "off" the restored programs; that is, return temporary ID segments to the operating system.

Program Description

Transfer file /TRANZ issues Restore Program (:RP) commands for programs TRANZ, CASDS, UNPKZ, EDITZ, and DBHIZ. It also prints a message reminding the user to set the system clock before beginning transcription. If any of the programs to be restored have not been previously offed, an FMGR 023 error will result. The user should issue a :TR command each time this happens.

After transcription, transfer file \TRANZ is used to off all the transcription programs, thus making their temporary ID segments available to the system.

Special Requirements

Transfer files /TRANZ and \TRANZ should be stored on logical unit 2 since the File Manager peripheral disk is not mounted during transcription.

Program Loading

There is no program loading required. The transfer files must, however, be stored on LU 2. If a copy of file /TRANZ is on the tape drive (LU 8) the following commands can be used to store it on disk:

```
:ST,8,/TRANZ::-2
```

If a copy of /TRANZ is to be transferred from a peripheral disk, one of the following commands can be used:

```
:ST,/TRANZ::-10,/TRANZ::-2
```

:ST,/TRANZ::300,/TRANZ::-2

The first command copies from logical unit 10, which is usually the peripheral disk, to logical unit 2, the system disk. The second command would copy the file from a disk with cartridge reference number (CR) 300 to LU 2. The above commands can also be used to copy file \TRANZ simply by changing the file name.

Program Operation

The transfer files /TRANZ and \TRANZ are run with the following File Manager commands, respectively:

:TR,/TRANZ

:TR,\TRANZ

9. Appendix A - Data Formats

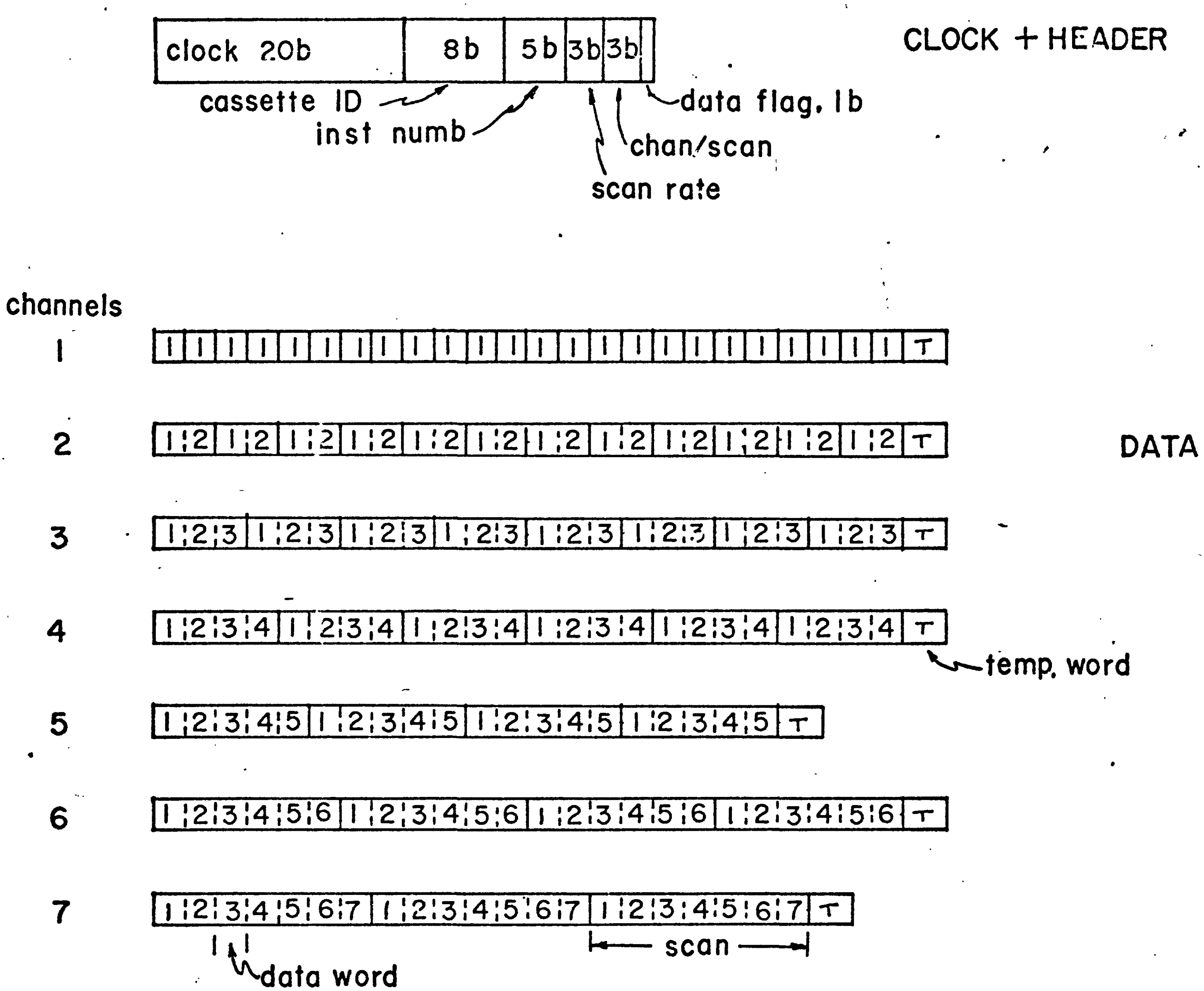
This appendix consists of three sections, each describing different data formats which are used during the data transcription process. The first section describes the data organization in the data logger and on the magnetic cassettes. The next section diskusses the organization of the data as it comes out of the cassette reader and goes into the computer. The last section describes the source-tape format. The source tapes are the final output from the transcription.

<u>Data Logger and Cassette Format</u>

The Sea Data Model 651-2 Data Logger assembles clock, header, and data information in a buffer with a capacity of 344 bits. (See Figure 9.1.) The clock and header are each 20 bits long, while the data occupies from 256 to 304 bits depending upon the number of channels per scan. The clock increments every half second. One clock increment is called a tick (2 ticks = 1 second). The header consists of two binary-coded decimal digits (eight bits total) representing the cassette ID number (0-99), a five-bit instrument number (1-31), a three-bit scan rate code (1-7), the number of channels per scan (1-7) represented by three bits, and a one-bit data flag. The sample interval in ticks is 2**(scan interval code). The data flag can be set by attaching a jumper on the back of the data logger control panel.

Data words are 12 bits long with up to 24 data words contained in a single cassette record. Data channels are scanned in turn a total of NSCAN times, where NSCAN=integer(24/NCHAN). A cassette record consists of a 20-bit clock word, 20 bits of header information, NSCAN's of data, a 16-bit temperature word. This entire record is written on the cassettes as four-bit characters preceded by a two-character preamble (0000 1111), and followed by a four-bit longitudinal parity character. The parity character provides odd

Figure 9.1 Data logger data format.



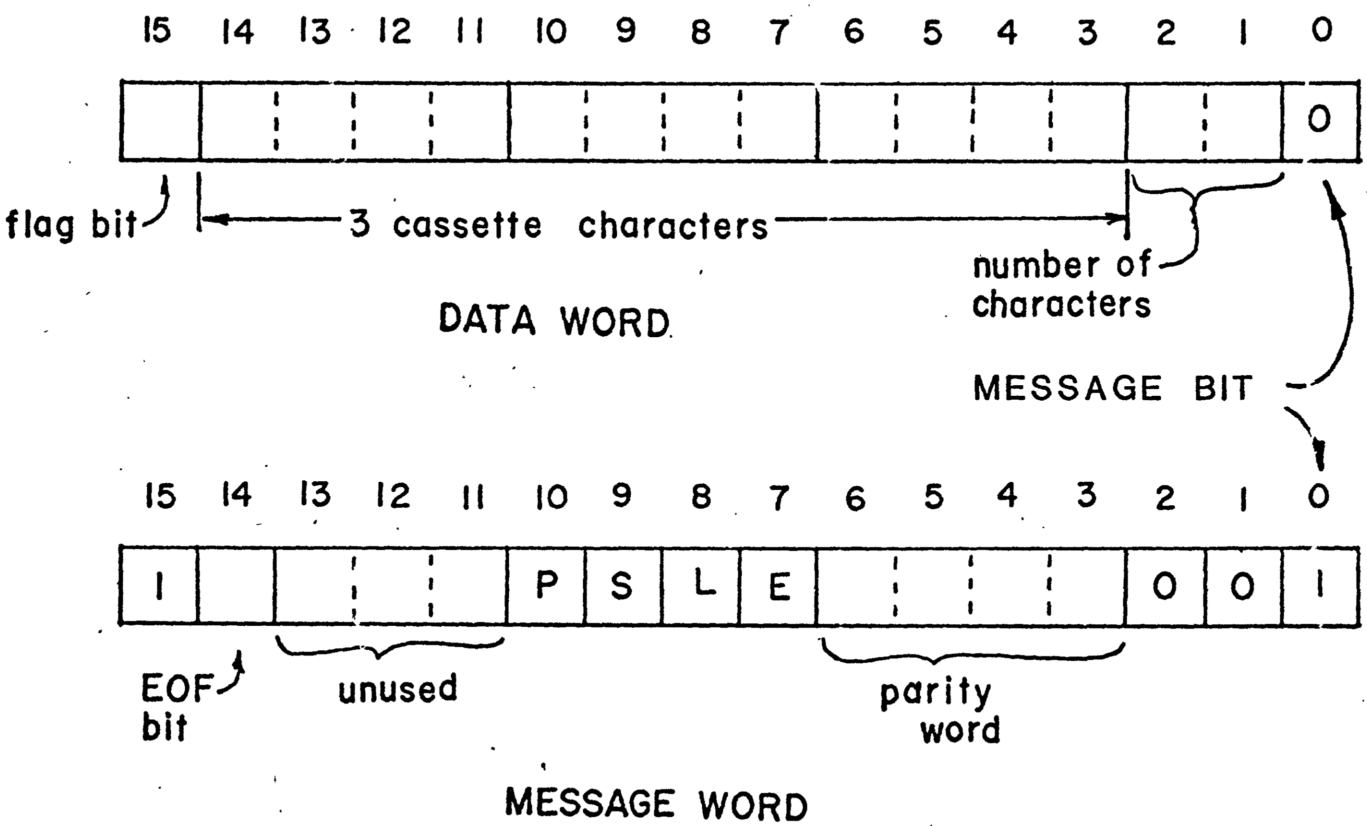
parity for each cassette track. An interrecord gap seven characters long is also written.

Cassette Reader Output

The Sea Data Model 12 Cassette Reader is used for transcribing the data cassettes described below. The data, in groups of three characters (12 bits) plus four additional bits supplied by the reader, is transmitted to the computer as a 16-bit word called a reader data word (See Figure 9.2). A reader data word contains a flag bit (bit 15), which is set to zero except for the last data word in a cassette record. The next 12 bits are cassette characters which are left justified if fewer than three characters are present. The next two bits indicate the number of cassette characters in the previous data field. The last bit is called the message bit. If the flag bit in this data word is turned on and the message bit is zero, the data word is the last one in that cassette record. If the flag and message bits are turned on, the word is called a message word.

The message word follows the last data word of each record. Bit 14 is reserved for use by end-of-file detection hardware in the reader (EOF=1). Bits 13, 12, and 11 are presently unused. Bits 10 through 7 indicate reader errors P, S, L, and E, respectively; the bits are set if the error condition exists. The P bit indicates a parity error, and the S bit indicates a short record caused by a severe data drop out. These two error indicators are the most reliable. The L bit is set when a low signal is detected by the reader, and indicates a possible data drop out. The E bit indicates excess data in a record, and can be caused by a cassette being erased quite well before recording. The E and L bits do not indicate definite data errors. Bits 6 through 3 indicate if the parity (odd) checking circuitry in the reader obtained the same result as that written by the recorder. If the parity

Figure 9.2 Cassette reader data format.



on a track is odd, as it should be, the corresponding bit is set. If a parity error exists on a track, the corresponding bit is cleared. Bits 2 and 1 of a message word are always zero.

Thirty-two cassette records are combined to form one disk record. These records are then unpacked and written onto magnetic tape using the format described in the next section. The table below shows the number of bits, characters, and words of the various data arrangements for different numbers of data channels.

Table 9.1 Record length of data in cassette records, unpacked disk records, and source tape records for different numbers of data channels.

nchan	nscan	bits/ cass rec	cass char/ rec	data wds/ rec(1)	tot wds/ cass rec(2)	wds/ init disk rec(3)	wds/ source tape rec
1	24	344	86	29	30	960	1024
2	12	344	86	29	30	960	1024
3	8	344	86	29	30	960	1024
4	6	344	86	29	30	960	1024
5	4	296	74	25	26	832	896
6	4	344	86	29	30	960	1024
7	3	308	77	26	27	864	928

(1) - (cassette characters + 1 parity character)/3

(2) - one word added for message word

(3) - 32 cassette records (subrecords) for one intermediate disk record or tape record

Source-Tape Format

Source tapes contain the transcribed data in an unpacked format. One file on a source tape represents one cassette image. Each file contains a header record of 128 words followed by data records. (See Figure 9.3) The last data record is followed by an end-of-file (EOF) mark, and the last file on a tape is followed by two EOF's. The description of the header words is given below.

Table 9.2 Source tape header record format

Word	Contents
1	Transcription version number formed from the version number of TRANZ, UNPKZ, EDITZ, and DBHIZ (1000*TRANZ + 100*UNPKZ + 10*EDITZ + DBHIZ)
2	Day of year of transcription
3	Year of transcription
4	Tape file number (0-32767)
5	1st and 2nd character of location code (ASCII)
6	3rd and 4th character of location code (ASCII)
7	Cassette ID number (0-99)
8	Instrument number (1-31)
9	Scan rate (0-7), NRATE (Sample interval=2**(NRATE-1) seconds)
10	Channels per scan (1-7), NCHAN
11	Clock reset time, hours
12	Clock reset time, minutes
13	Clock reset time, day
14	Clock reset time, month
15	Clock reset time, year
16	Clock off time, hour
17	Clock off time, minute
18	Clock off time, day
19	Clock off time, month
20	Clock off time, year

Table 9.2 Continued

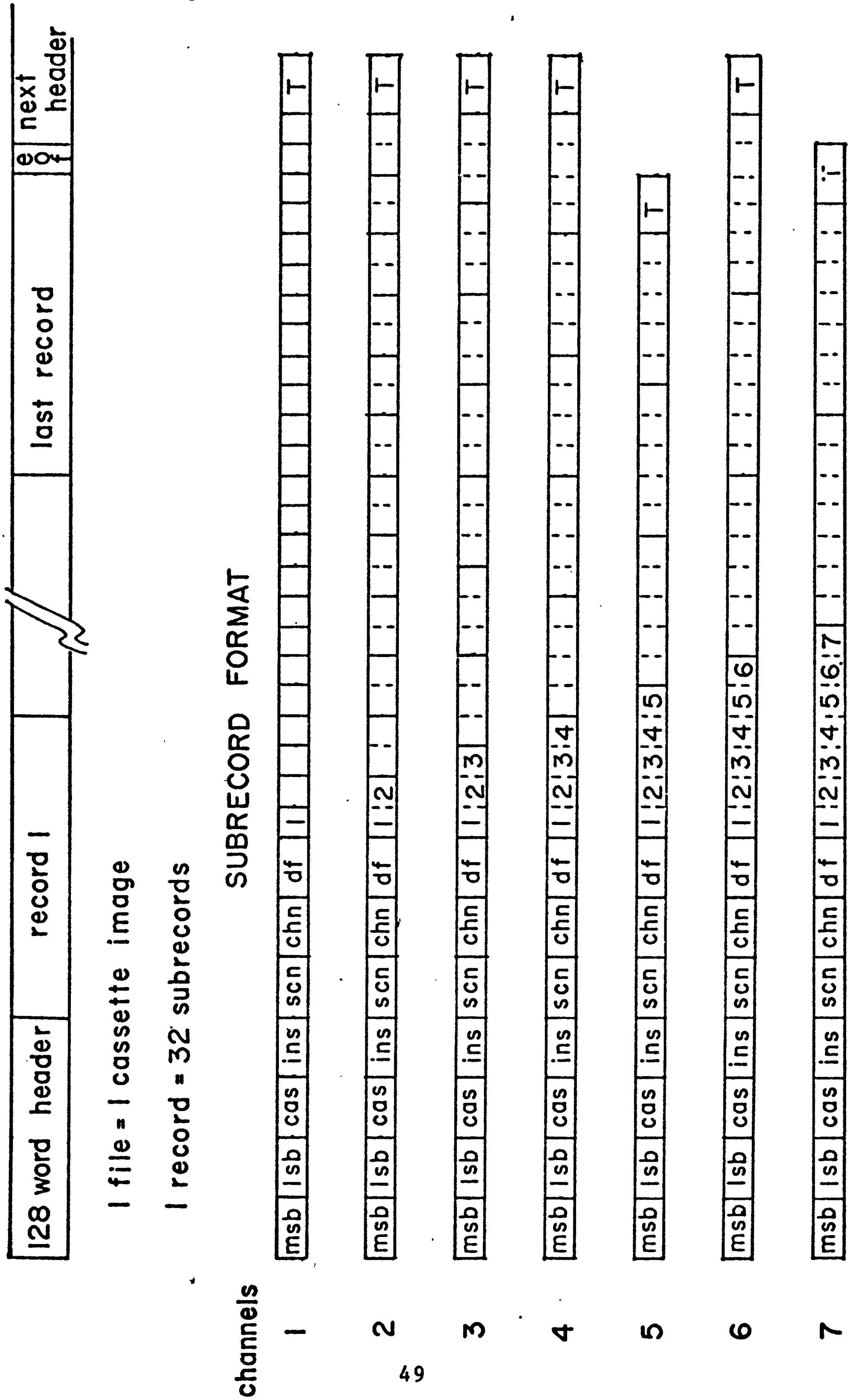
Word	Contents
21	Stop watch time between last record and next WWV minute mark, minute
22	Stop watch time, second
23	Stop watch time, tenths of second
24	Number of words per cassette record
25	Number of cassette records per disk record (always 32)
26	Number of words per tape record, NBUFL
27-51	Comment field (50 ASCII characters)
52	Number of words per subrecord, NWORD (NWORD=NSCAN*NCHAN+8)
53	Number of scans per subrecord, NSCAN (NSCAN = integer(24/NCHAN)
54	Hx gain in nT/2048 counts (Value of 0 indicates a default of 1000 nT/2048 counts.)
55	Hy gain
56	Hz gain
57	Ex gain, >0 north end (+), <0 south end (+)
58	Ey gain, >0 east end (+), <0 west end (+)
59	Ex line length in meters
60	Ey line length in meters
61-68	Currently unused. N.B. Some of these words are used in the disk record header when data segments are extracted by program SLECT which is not described in this report.

Each source tape record consists of 32 subrecords. The format of the subrecords is shown in Figure 9.3. They consist of seven words of subheader followed by data words and a temperature word. The first 2 words of the subheader are the 5 most significant clock bits, and the 15 least significant clock bits. The clock value is reconstructed by the formula

$$CLOCK = 32768*MSB + LSB.$$

The user should keep in mind that the clock rolls over at 2**20 = 1,048,576.

Figure 9.3 Source tape format.



The clock values are in one-half second increments called ticks. The next word is the cassette ID number, which is followed by the instrument number. A negative instrument number indicates a data break, i.e., there is data missing between this and the previous subrecord. A zero value means there is no more data in the record and the file. NRATE, the scan rate, and NCHAN, the number of channels per scan, are next. The last word in the subheader is the data flag word.

The data flag word is shown in Figure 9.4. Bit 0 is the data flag bit, which is set by attaching a jumper on the back of the cassette recorder

15 9	8	7	6	5	4 1	0
IFLGE	P	S	L	E	Parity Word	DF

Figure 9.4 Format of subheader data flag word.

control panel. It presently is not used. Bits 8-5 are the reader message bits P, S, L, and E, and bits 4-1 are the parity word for the subheader. Bits 15-9 contain the number of missing subrecords between the present and previous subrecord.

The data follows the subheader with NCHAN channels of data scanned NSCAN times. The data words are 12 bits long, right justified in a 16-bit word. The last word in a subrecord is the temperature word. The clock value in the subheader and the temperature word correspond to the last data scan in the subheader.

The data logger is usually wired to scan Hx, Hy, and Hz magnetic channels, and then scan the Ex and Ey electric field channels. The magnetic field value is related to the number of counts in the data word by the formula

$$\text{magnetic field(nT)} = \text{GAIN*(COUNTS-2048)}.$$

where the GAIN is gotten from the header and the COUNTS are the data word. Note that the data is recorded in offset binary, i.e., numbers between 2049 and 4095 are positive, while those from 0 to 2047 are negative. The electric field is related to the number of counts by the formula

$$\text{electric field (mV/km)} = \frac{\text{4882.813*(COUNTS-2048)}}{\text{GAIN*LINE\_LENGTH(m)}}$$

From the temperature word and apparent temperature can be obtained

$$T_a(^{o}C) = \frac{72\text{*COUNTS} - 6}{4096}$$

This has an accuracy of $\pm 0.25^{o}C$ over the range 10 - 45 ^{o}C. To obtain the true temperature outside of this range, the following correction terms are added.

Table 9.3 Temperature word correction constants

$T_a(^{o}C)$	Correction (^{o}C)
66	+9.0
60	+4.3
55	+2.5
10	-0.25
5	-1.5
0	-3.5
-6	-9.0

All data words are 16 bits long to allow easy access to data without need for unpacking.

10. Appendix B - User's Guide

This appendix show examples of output and user responses for the different transcription programs, and gives some explanation of what the output means. It is not intended to be an exhaustive description of the functioning of the programs. The reader is encouraged to read the Program Description sections and peruse the program listings for answers to detailed questions. Table 10.1 gives a summary of the function of the programs used in the transcription process.

The format used in this section is figures of actual transcription output keyed with circled numbers. The numbers correspond to more detailed comments in the body of the text.

We will diskuss the output in Figure 10.1 first. The numbers below correspond to the circled number in the figure.

1. This is the WELCOM message which is printed when the operating system is booted up.
2. This command is issued to execute the File Manager instructions stored in file /TRANZ which restores the transcription programs.
3. These are the commands stored in /TRANZ which are printed before they are executed.
4. The user, believing the maxim "The early bird catches the worm" is setting the system clock to 6:15 on the 29th day of 1980.
5. Program TRANZ is run.
6. Due to the early hour of the day, the user has forgotten to remove the peripheral disk before running TRANZ. A response of "NO" is given to allow this task to be done. Undoubtedly the user would do well to heed B. Franklin's advice of "Early to bed......".

Table 10.1 Summary of transcription operation and program functions. Numbers in parentheses are figures to refer to for examples of output.

Program	Description
TRANZ	Controls overall transcription process including input of transcription parameters and scheduling of subprograms. Writes transcribed data to tape. (10.1, 10.2)
CASDS	Transfers data from cassette to disk. (10.2)
UNPKZ	Unpacks data on disk and does cassette record length editing. (10.3, 10.4)
EDITZ	Edits data for inconsistent header information. Checks clock incrementation. Notifies user of and/or removes data with parity errors. Tallies other reader errors. (10.4, 10.5, 10.6, 10.7)
DBHIZ	Prints a summary of data breaks and a histogram of data values. (10.7, 10.8, 10.9)

Figure 10.1 Terminal output from boot up, program restoration, and aborted transcription attempt.

```
SET TIME
 :SV,4
TE,*****
TE,*****      GMV PROCESSING SYSTEM
TE,*****      RTE-II REV. 1913
TE,*****      GENERATED 13 JULY 1979
TE,*****
TE,
TE,        DO NOT PURGE ANY FILES WHICH ARE NOT YOUR OWN!
TE,
 :TR
:TR,/TRANZ
 :RP,TRANZ
 :RP,CASDS
 :RP,UNPKZ
 :RP,EDITZ
 :RP,DBHIZ
 :SV,4
TE,*** BE SURE TO SET SYSTEM CLOCK BEFORE TRANSCRIBING ***
 :TR
:SYTM,198Ø,29,6,15
:RU,TRANZ

 IS A SCRATCH PACK MOUNTED ON LU #1Ø? (YE OR NO) NO
  TRANZ : STOP  ØØØ3
:DC,-1Ø
LAST TRACK  Ø2Ø2
```

7. The File Manager peripheral disk pack is dismounted from LU 10.

8. The peripheral disk is replaced with a scratch disk.

The false start taken care of, we move on to a second try at the transcription (see Figure 10.2).

1. TRANZ is run again and the user has decided to save data with parity errors.

2. Transcription parameters are input.

3. An input error has been made. The user typed 15, then decided to remove the 5 so the backspace (BS) key was pressed. This action printed the underscore. The user then decided to delete the entire input using the DELETE key. The backslash was printed and the paper advanced to the next line. The value of "1" is now input.

4. The rest of the transcription parameters are input. If the prompt is ever repeated, it means an unallowable parameter has been supplied.

5. Program CASDS is scheduled after this message has been printed. The user loads the cassette and sets the reader to the indicated number of characters per record and presses the READ button. The levels of the signal on the 4 data channels should be monitored, and the gain for the corresponding channels adjusted to keep the PERCENT of REFERENCE meter between 90 and 110. When the signal level drops to zero, or you do not want to read anymore data, press bit 15 of the display register.

Figure 10.2 Terminal output from transcription parameter input and cassette reading.

```
:RU,TRANZ

IS A SCRATCH PACK MOUNTED ON LU #10? (YE OR NO) YE

SAVE DATA WITH PARITY ERRORS? (YE OR NO) YE          1

PROGRAM TRANZ  VERSION -  5  DAY: 29 YEAR:1983


TAPE FILE #?  (<) TO STOP) 615
LOCATION CODE?  (4 CHAR.) EUG                        2
CASSETTE ID #?  (0-99) 15
INSTRUMENT NUMBER?  (1-31) 15                        3
SCAN RATE? (0-7) 3
CHANNELS/SCAN?  (1-7) 3
MAGNETOMETER GAIN IN GAMMAS/2048 COUNTS
HX=  500 HY=  500 HZ=  500
IF OK TYPE /'S, IF NOT TYPE NEW VALUES
HX    HY    HZ
/ / /                                                4
RESET TIME?  (HR MIN DAY MON YR) 13 15 09 07 1979
OFF TIME?    (HR MIN DAY MON YR) 17 08 19 07 1979
STOP WATCH?  (MIN SEC TENTHS) 0 58 2
COMMENTS?  (50 CHARACTERS)
RETRAIS 4 OF EUGENE, OR AIRPORT
SET READER CHARACTERS/RECORD=86.  PRESS READ.        5

CASDS COMPLETED
CASRC= 18334 DISRC=   591 BADRC= 14987 COMST=000000B LSTRK=    98   6
```

6. After bit 15 is set, TRANZ prints this message indicating CASDS is done. The message displays the number of cassette records read (CASRC), the number of disk records written (DISRC), the number of cassette records which had a reader error bit set (BADRC), the completion status of the transcription (COMST), and the current value of the track write address (LSTRK).

Program TRANZ also writes a summary of the input parameters and the message described in item 5 (above) on the line printer.

Below is a summary of the input parameters furnished by the user to program TRANZ.

TAPE FILE	-Number assigned to uniquely identify this cassette image (0-32767). A value less than 0 will terminate TRANZ.
LOCATION CODE	-Code name of the field station. May be up to 4 characters long. If the location code is shorter, it is left justified.
CASSETTE ID	- Cassette identification number dialed in on the data logger (0-99).
INSTRUMENT NUMBER	- Number of instrument which is set by jumpers on the back of the data-logger control panel (1-31).
SCAN RATE	- Set on data-logger front panel to control the sample interval (0-7). The sample interval is equal to 2**(SCAN RATE - 1) seconds.
CHANNELS/SCAN	- Number of data channels sampled by the data logger (1-7). The temperature channel is not included in this count.

MAGNETOMETER GAIN -The number of nanoteslas (nT or gammas) per 2048 counts. For E.D.A. FM-100-B magnetometers straight from the factor this number is 1000. These values are stored in file MAGAIN. Values can be changed by typing new values, or retained by typing zeros. See Section 7 for information about modifying or listing the gain files.

For instruments recording 5 channels of data the following four questions are also asked.

EX GAIN - North-south telluric channel gain. Positive if the (+) electrode is the north end of the dipole, and negative if the (+) electrode is the south end of the dipole.

EX LENGTH -Length of the north-south electric field line in meters.

EX GAIN - East-west telluric channel gain. Positive if the (+) electrode is the east end of the dipole, and negative if the (+) electrode is the west end of the dipole.

EY LENGTH - Length of the east-west electric field line in meters.

The remaining questions are asked for all transcriptions.

RESET TIME - The hour, minute, day, month, and year the data-logger clock was reset to zero.

OFF TIME - The hour, minute, day, month, and year of the nearest recordable minute mark <u>after</u> the data logger stopped recording.

STOP WATCH - The minutes, seconds, and tenth's of seconds between the last data-logger record and the OFF TIME.

COMMENTS - Up to 50 character of user-furnished comments. This will appear in the header and is often printed by other processing programs.

The next program scheduled by TRANZ is UNPKZ. Almost all output from UNPKZ is displayed on the line printer (see Figure 10.3). Whenever UNPKZ finds a cassette record with the wrong number of words in it, an error message is printed. Two of the three possible error messages are shown.

1. While processing input disk record 60 (IDSRC) a record with 58 words (IEXIT) was found when a length of 30 words (NWCAS) was expected. The 58 word record started at location 451 (IPT+1) of the input buffer. The only reader error detected was an excess data error (PSLE = 0001). Since the record length was long, a partial delete was done, and the last 30 words were saved for further processing.

2. In this case a short record with only 29 words was encountered. The reader indicated that there was a parity and excess data error (PSLE = 1001). The entire record was deleted from the data set.

3. After 100 error messages are printed, the user is asked if any more messages should be printed (see Figure 10.4). A response of "NO" was given for this example on the 104th input disk record.

Figure 10.3 Examples of UNPKZ wrong-length record messages.

```
UNPKZ RECORD LENGTH ERRORS
 IDSRC=   46 IPT= 538 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   46 IPT= 657 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED  1
 IDSRC=   46 IPT= 746 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   60 IPT= 450 IEXIT=   58 NWCAS=30 PSLE=0001 LONG RECORD  - PARTIAL DELETE
 IDSRC=   63 IPT= 240 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   63 IPT= 299 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   63 IPT= 388 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   63 IPT= 717 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   64 IPT= 630 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   64 IPT= 659 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   64 IPT= 688 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   64 IPT= 747 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   64 IPT= 866 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   64 IPT= 925 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   66 IPT=  60 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   66 IPT= 269 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   66 IPT= 868 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   72 IPT= 210 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   72 IPT= 509 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   72 IPT= 658 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   77 IPT= 420 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   77 IPT= 449 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   77 IPT= 688 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   77 IPT= 777 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   79 IPT= 300 IEXIT=   29 NWCAS=30 PSLE=0001 SHORT RECORD - DELETED
 IDSRC=   79 IPT= 329 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   80 IPT= 150 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   80 IPT= 179 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   81 IPT= 780 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   82 IPT= 150 IEXIT=   29 NWCAS=30 PSLE=0001 SHORT RECORD - DELETED
 IDSRC=   83 IPT=  60 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED  2
 IDSRC=   83 IPT= 119 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   83 IPT= 178 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   85 IPT= 510 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   85 IPT= 599 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   86 IPT= 270 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   86 IPT= 329 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   92 IPT= 510 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   92 IPT= 539 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   92 IPT= 598 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   92 IPT= 867 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   95 IPT=  90 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   97 IPT= 210 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   97 IPT= 359 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   97 IPT= 598 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=   97 IPT= 897 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=  100 IPT= 900 IEXIT=   29 NWCAS=30 PSLE=0001 SHORT RECORD - DELETED
 IDSRC=  101 IPT=  90 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=  102 IPT= 360 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=  102 IPT= 479 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=  104 IPT= 690 IEXIT=   29 NWCAS=30 PSLE=1001 SHORT RECORD - DELETED
 IDSRC=  104 IPT= 719 LEFT= 242  ERROR REPORTING TERMINATED  3
UNPKZ COMPLETED: NDSRC=  591 JDSRC=  575 NWLRC=  534
                 NCDEL=  529 NPDEL=    5                           4
                 PE= 1081 SH=    0 LO=     0 EX=14685 TOTAL=15766
```

4. UNPKZ has completed processing the data and has written this message on the line printer and terminal. There were a total of 591 input disk records (NDSRC) processed, 575 disk records output (JDSRC), and 534 wrong-length cassette records (NWLRC). Of these wrong-length cassette records, 529 were completely deleted (NCDEL), while 5 were partially deleted (NPDEL). Among the cassette records which were retained and written to disk, 1081 had parity errors (PE), there were no short (SH) or low signal (LO) errors, and 14,685 excess data (EX) records for a total of 15,766 errors.

There is a third error message which is printed when no end of record (EOR) is detected in the input buffer. This condition results in a complete delete of the data in the buffer. The message that is printed is similar to the complete delete message described above.

Figure 10.4 shows the terminal output from UNPKZ and the beginning of EDITZ.

1. This is the message advising the user that 100 error messages have been printed by UNPKZ. The user can stop the output by responding "NO". A response of "YE" will let 100 more messages be printed before the question is asked again. This feature was mentioned in item 3 above.
2. This is the terminal message corresponding to item 4 above.
3. After UNPKZ is completed, program EDITZ begins by getting the clock properly initialized. The message indicates there is a clock-incrementation error on input disk record 1 (IDSRC) on the clock value in location 1 (IPTC+1). The next output disk record will be number 1 (JDSRC), and the next output clock value will go into location 5 (JPTC+1) of the output clock buffer. The subrecord will be number 1 (JSRC) of output record number 1 (JDSRC).

Figure 10.4 Terminal output from program UNPKZ and the beginning of program EDITZ.

```
ERRORS EXCEED PRINTING LIMIT OF    1ØØ
CONTINUE REPORTING ERRORS? (YE OR NO) NO                          (1)
UNPKZ COMPLETED: NDSRC=  591 JDSRC=  575 NWLRC=  534
                 NCDEL=  529 NPDEL=    5                          (2)
                 PE= 1Ø81 SH=    Ø LO=    Ø EX=14635 TOTAL=15766
 UNPKZ : STOP  ØØØØ
                                                                  (3)
CLOCK ERROR: IDSRC=   1 IPTC= Ø JDSRC=   1 JPTC= 4 JSRC= 1

JSRC  CLKA      DTA      CLKC      DTC      MSRC
29       Ø.      -         Ø.       -        Ø
3Ø       Ø.      Ø.        Ø.       Ø.       Ø
31       Ø.      Ø.        Ø.       Ø.       Ø
32       Ø.      Ø.        Ø.       Ø.       Ø                    (4)
 1     3Ø8.    3Ø8.       ?         -        1
 2     372.     64.       ?         -        Ø
 3     436.     64.       ?         -        Ø
 4     564.    128.       ?         -        1

DT=  64.   BEST CLOCK ESTIMATE=      128.
USE BEST CLOCK ESTIMATE? (YE OR NO) NO                            (5)
CORRECTED CLOCK VALUE? 3Ø8
```

4. This output display gives a picture of the actual and corrected clock values and missing cassette record numbers before and after the clock value which is being processed. The first column (JSRC) indicates the output subrecord number where the data will go. This number ranges between 1 and 32. CLCKA refers to the actual, uncorrected clock values, and DTA is the difference between the actual clock value on that and the previous line. For example, on the line with JSRC equal to 2, the actual clock value is 372. This is 64 ticks greater (DTA) than the actual clock value on the previous line. The columns labeled CLKC and DTC correspond to the corrected clock and its difference. The clock value currently being processed is on the first line with a question mark. Notice that from this line forward in time there are no corrected clock values (CLKC) or differences (DTC) since the data have not been processed. The right-hand column (MSRC) indicates the number of missing records. Lines corresponding to corrected clock values are in agreement with the value of DTD, i.e., DTC=DT*(MSRC+1). Values corresponding to uncorrected clock values may have inaccuracies as described in Sections 4 and 5 of this report.

5. The user is told the correct clock increment (DT) if no data were missing and "the best clock estimate." This value is equal to the previous corrected clock value plus DT times the current MSRC plus 1. For the first clock value of a transcription, this is usually not a good estimate because the first cassette record is rarely written at DT ticks after the clock reset. In this example, the user has decided not to use the estimate, but has input a value of 308. This corresponds to the first actual clock value, which is usually the best value to use unless the first clock value is obviously wrong due to a parity error. If there is reason to believe that all of the clock values are shifted by a constant

amount, a correcting bias can be added at this point. This situation might arise if the clock reset time was incorrectly noted.

More terminal output from program EDITZ is shown in Figure 10.5.

1. EDITZ has a parity error limit which is set to 10 when data with parity errors are being saved. When this limit is exceeded, a message is printed asking if processing should be continued. If it is, the differential parity error limit is doubled. This example shows a total of 4 limit doublings. The message contains the number of input disk records (IDSRC), the clock pointer for the input record (IPTC), the total number of disk records to be processed (NDSRC), and the total number of parity errors encountered.
2. When inconsistent header errors exceed a set limit (5), a message is printed showing the recorded header and the corrected header. The latter values are based on the input parameters supplied to TRANZ by the user. These values correspond to words 3, 4, 5, and 6 of the subheader. They are respectively: the cassette ID number, the instrument number, the scan rate, and the number of channels per scan. The output is displayed as octal numbers. Also displayed are the values of IDSRC, IPTC, and NDSRC, which were described in item 1 above. If the user answers "YE" to the question "CONTINUE PROCESSING", no more inconsistent header errors will be printed. A "NO" response will stop processing of the data and return to TRANZ. Before answering the question, the user should check to be certain a typographical error was not made when the subheader values were input.
3. This is another example of a clock incrementation error. The clock value in question will go into subrecord 13 of output disk record 281. The value of DTA (192) is 3 times DT (64), and there are 2 missing records. These numbers have the proper relationship <u>i.e.</u>, DTA=DT*(MSRC+1), so the

Figure 10.5 Terminal output showing parity error and inconsistent header error overflows. Also shown is a clock error correction.

```
PARITY ERRORS EXCEEDED DIFFERENTIAL LIMIT (  1Ø)
 IDSRC=  93 IPTC=33 NDSRC= 575 TOTAL PR ERRORS=    11     (1)
CONTINUE PROCESSING? (YE OR NO) YE

PARITY ERRORS EXCEEDED DIFFERENTIAL LIMIT (  2Ø)
 IDSRC= 145 IPTC=18 NDSRC= 575 TOTAL PR ERRORS=    32     (1)
CONTINUE PROCESSING? (YE OR NO) YE

HEADER ERRORS EXCEED DIFFERENTIAL COUNT LIMIT (  5)
RECORDED  HEADER: ØØØØ43B ØØØØØ1B ØØØØ13B ØØØ4Ø3B *
CORRECTED HEADER: ØØØØ17B Ø4ØØØ1B ØØØØ13B ØØØ113B         (2)
IDSRC= 211 IPTC=17 NDSRC= 575 TOTAL IC ERRORS=  6
CONTINUE PROCESSING? (YE OR NO) YE

PARITY ERRORS EXCEEDED DIFFERENTIAL LIMIT (  4Ø)
 IDSRC= 213 IPTC=24 NDSRC= 575 TOTAL PR ERRORS=    73     (1)
CONTINUE PROCESSING? (YE OR NO) YE
CLOCK ERROR: IDSRC= 281 IPTC=15 JDSRC= 281 JPTC=15 JSRC=13

JSRC  CLKA         DTA       CLKC        DTC      ISRC
  9   591155.       -       591156.       -        Ø
 10   591284.      128.     591284.     128.       1
 11   591348.       64.     591348.      64.       Ø
 12   591412.       64.     591412.      64.       Ø      (3)
 13   5916Ø4.      192.        ?          -        2
 14   624436.    32832.        ?          -        Ø
 15   591732.   -32704.        ?          -        Ø
 16   591860.      128.        ?          -        1
DT=   64.    BEST CLOCK ESTIMATE=  5916Ø4.
USE BEST CLOCK ESTIMATE? (YE OR NO) YE

PARITY ERRORS EXCEEDED DIFFERENTIAL LIMIT (  3Ø)
 IDSRC= 293 IPTC=15 IDSRC= 575 TOTAL PR ERRORS=   154     (1)
CONTINUE PROCESSING? (YE OR NO) YE
```

best clock estimate is used.

Program EDITZ also prints error messages on the line prnter. Some examples of this output are shown in Figure 10.5.

1. Whenever the user is asked to help correct a clock incrementation error, a message like this one is printed. This message corresponds to item 4 in Figure 10.4. The input disk record, input clock pointer, and actual clock value are displayed.
2. This is a message from an excess data error occurring on input disk record 1 with a pointer value of 3, and an actual clock value of 564 ticks. Only the first 5 occurrences of this error are reported. The data are always retained.
3. The clock ambiguity message is printed whenever secondary clock differencing was used to correct the clock, and the number of missing records had to be used to locate the middle clock value. See Section 5 which describes the procedure used by program EDITZ for a complete discussion of the parameters printed.
4. A parity error was encountered on input disk record 85 with clock pointer value of 23, and an actual clock value of 179,060. The parity error occurred in the third data track. This corresponds to the digit in PWORD which is not equal to 1.
5. This is an incorrect header error message. The value of HWORD indicates there is an inconsistency in the first word compared (word 3 of the subheader).
6. After 100 parity error messages are printed, this message is written to advise the user that no more parity error messages will be printed.
7. This message is printed when EDITZ is done processing data. A total of 574 (NDSRC) disk records were written. These records contained 29

Figure 10.6 Examples of printer output from program EDITZ.

```
EDITZ ERROR REPORT
CLOCK  INCREMENTATION ERROR : IDSRC=   1 IPTC= 0 CLKAI=     308.   1
EXCESS DATA  - DATA RETAINED: IDSRC=   1 IPTC= 3 CLKAI=     564.   2
CLOCK AMBIGUITY - IMSR1 USED: JDSRC=   1 JSRC= 4 I1=     2 I2=      3 IMSR1=    1 IMSR2=    0
                                                                    3
EXCESS DATA  - DATA RETAINED: IDSRC=   1 IPTC= 5 CLKAI=     692.
EXCESS DATA  - DATA RETAINED: IDSRC=   1 IPTC= 6 CLKAI=     756.
EXCESS DATA  - DATA RETAINED: IDSRC=   1 IPTC= 7 CLKAI=     820.
EXCESS DATA  - DATA RETAINED: IDSRC=   1 IPTC= 8 CLKAI=     884.

PARITY ERROR - DATA RETAINED: IDSRC=  85 IPTC=23 CLKAI= 179060. PWORD=1101   4
PARITY ERROR - DATA RETAINED: IDSRC=  88 IPTA=14 CLKAI= 184628. PWORD=1101
INCORRECT HEADER - CORRECTED: IDSRC=  88 IPTC=14 CLKAI= 184628. HWORD=1000   5
            RECORDED  HEADER: 000043B 000001B 000003B 000003B
           CORRECTED  HEADER: 000017B 000001B 000003B 000003B
CLOCK AMBIGUITY - IMSR1 USED: JDSRC=  89 JSRC=31 I1=      3 I2=      5 IMSR1=   2 IMSR2=    1
CLOCK AMBIGUITY - IMSR1 USED: JDSRC=  90 JSRC= 8 I1=      2 I2=      3 IMSR1=   1 IMSR2=    0
CLOCK AMBIGUITY - IMSR1 USED: JDSRC=  92 JSRC=13 I1=      2 I2=      3 IMSR1=   1 IMSR2=    0
PARITY ERROR - DATA RETAINED: IDSRC=  93 IPTC=30 CLKAI= 196212. PWORD=1101

PARITY ERROR - DATA RETAINED: IDSRC= 264 IPTC=21 CLKAI= 556660. PWORD=0111
PARITY ERROR - DATA RETAINED: IDSRC= 265 IPTC= 9 CLKAI= 557940. PWORD=0111
** NO MORE PARITY ERROR MESSAGES WILL BE PRINTED **   6
CLOCK AMBIGUITY - IMSR1 USED: JDSRC= 266 JSRC=18 I1=  -511 I2=      2 IMSR1=   0 IMSR2=    0
CLOCK AMBIGUITY - IMSR1 USED: JDSRC= 266 JSRC=32 I1= -8191 I2=      2 IMSR1=   0 IMSR2=    0

CLOCK AMBIGUITY - IMSR1 USED: JDSRC= 557 JSRC=31 I1=     -3 I2=      2 IMSR1=   0 IMSR2=    0
CLOCK AMBIGUITY - IMSR1 USED: JDSRC= 561 JSRC= 1 I1=      2 I2=      3 IMSR1=   1 IMSR2=    0
CLOCK AMBIGUITY - IMSR1 USED: JDSRC= 571 JSRC= 6 I1=     -3 I2=      2 IMSR1=   0 IMSR2=    0
CLOCK AMBIGUITY - IMSR1 USED: JDSRC= 574 JSRC= 6 I1=      2 I2=      3 IMSR1=   1 IMSR2=    0
EDITZ COMPLETE: NDSRC= 574
IC=    29 PR=   566 SH=     0 LO=     0 EX= 13761 CI=     10 TOTAL= 14366   7
```

inconsistent header (IC) errors, 566 parity (PR) errors, no short (SH) or low-signal (LO) errors, 13,761 excess data (EX) errors, and 10 clock incrementation (CI) errors for a total of 14,366 errors. This message is also printed on the terminal (see Figure 10.7).

Figure 10.7 shows terminal output from EDITZ, DBHIZ, and TRANZ.

1. This is the completion message of EDITZ which is described in item 7 above.

2. These STOP messages are written by EDITZ and DBHIZ when they have terminated.

3. This message is written by TRANZ after it has written the transcribed data onto magnetic tape. In this example, version 5 of TRANZ, version 2 of UNPKZ, version 3 of EDITZ, and version 1 of DBHIZ were used. Five hundred and seventy-four (574) data records were written to tape. The record count does not include the 128-word header record.

Program DBHIZ is the last program run before the data is written onto magnetic tape. It requires no operator intervention. The data-break summary from DBHIZ is shown in Figure 10.8. Whenever a clock difference which is not equal to DT is encountered, a message is written. The first two columns tell the record and subrecord of the offending data. The clock value of the previous and present subrecord are displayed along with their difference modulo 2**20. The next column contains the number of missing subrecords. If the clock jump was a data break (word 2 of the subheader negative) the letters "DB" are printed. Any output lines which are not marked "DB" should be considered suspicious, and are probably the result of faulty processing by EDITZ. The modular difference should be DT times the number of missing cassette records plus one, except for the first subrecord.

Figure 10.7 Terminal output for completion of EDITZ, DBHIZ, and writing of data to magnetic tape by TRANZ.

```
EDITZ COMPLETE: NDSRC= 574
IC=      29 PR=    536 SH=        0 LO=        0 EX= 13761 CI=      10 TOTAL= 1433[illegible]   1
 EDITZ : STOP   0000  } 2
 DBHIZ : STOP   0000  }
VERSION=5231    574 RECORDS WRITTEN  ← 3
```

Figure 10.8 Output from DBHIZ showing data-break analysis.

DATA BREAK SUMMARY: DT= 64.

REC	SUB REC	LAST CLOCK	PRESENT CLOCK	MODULAR DIFF	MISS SREC	
1	1	0.	308.	308.	1	DB
1	4	436.	564.	128.	1	DB
2	15	3252.	3380.	128.	1	DB
3	19	5620.	5748.	128.	1	DB
3	32	6516.	6644.	128.	1	DB
4	14	7476.	7668.	192.	2	DB
4	15	7668.	7796.	128.	1	DB
5	20	10100.	10228.	128.	1	DB
5	21	10228.	10548.	320.	4	DB
8	8	15796.	15924.	128.	1	DB
8	13	16180.	16308.	128.	1	DB
8	14	16308.	16500.	192.	2	DB
13	24	27316.	27444.	128.	1	DB
13	27	27572.	27700.	128.	1	DB
13	30	27828.	27956.	128.	1	DB
14	30	29940.	30068.	128.	1	DB
15	30	32052.	32244.	192.	2	DB
17	23	35828.	35956.	128.	1	DB
18	16	37492.	37620.	128.	1	DB
18	18	37684.	37812.	128.	1	DB
20	10	41332.	41460.	128.	1	DB
20	15	41716.	41844.	128.	1	DB
22	25	46516.	46708.	192.	2	DB
22	28	46836.	46964.	128.	1	DB
24	3	49396.	49524.	128.	1	DB
24	4	49524.	49652.	128.	1	DB
25	10	52020.	52212.	192.	2	DB
27	32	57652.	57844.	192.	2	DB
29	26	61492.	61620.	128.	1	DB
30	2	62068.	62196.	128.	1	DB
30	5	62324.	62516.	192.	2	DB
30	13	62964.	63156.	192.	2	DB
32	7	66804.	66932.	128.	1	DB
34	3	70708.	70836.	128.	1	DB
34	9	71156.	71284.	128.	1	DB
34	13	71476.	71604.	128.	1	DB
44	26	92852.	92980.	128.	1	DB
44	32	93300.	93428.	128.	1	DB
45	4	93620.	93748.	128.	1	DB
45	7	93876.	94004.	128.	1	DB
45	9	94068.	94196.	128.	1	DB
58	30	122100.	122228.	128.	1	DB
61	22	127796.	127924.	128.	1	DB
61	23	127924.	128052.	128.	1	DB
61	25	128116.	128244.	128.	1	DB
62	3	128820.	128948.	128.	1	DB
62	31	130676.	130932.	256.	3	DB
62	32	130932.	131060.	128.	1	DB
63	3	131188.	131316.	128.	1	DB
63	4	131316.	131444.	128.	1	DB
64	6	133556.	133684.	128.	1	DB
64	12	134004.	134132.	128.	1	DB
64	31	135284.	135412.	128.	1	DB
70	8	146164.	146292.	128.	1	DB
70	17	146804.	146932.	128.	1	DB
70	21	147124.	147252.	128.	1	DB
75	13	156916.	157108.	192.	2	DB

Figure 10.9 Histogram-analysis output from program DBHIZ.

HISTOGRAM SUMMARY

CHANNEL=1 NTOT= 146920. 1
MIN= 88 MAX=4041 FMIN= -478.5 FMAX= 486.6 2

BIN	MIN	MAX	FMIN	FMAX	N	%
1	0	127	-500.0	-469.0	1.	.0
2	128	255	-468.7	-437.7	24.	.0
3	256	383	-437.5	-406.5	0.	0.0
4	384	511	-406.2	-375.2	0.	0.0
5	512	639	-375.0	-344.0	0.	0.0
6	640	767	-343.7	-312.7	0.	0.0
7	768	895	-312.5	-281.5	0.	0.0
8	896	1023	-281.2	-250.2	0.	0.0
9	1024	1151	-250.0	-219.0	23.	.0
10	1152	1279	-218.7	-187.7	8.	.0
11	1280	1407	-187.5	-156.5	1.	.0
12	1408	1535	-156.2	-125.2	0.	0.0
13	1536	1663	-125.0	-94.0	3.	.0
14	1664	1791	-93.7	-62.7	8.	.0
15	1792	1919	-62.5	-31.5	856.	.6
16	1920	2047	-31.2	-.2	35599.	24.2
17	2048	2175	0.0	31.0	87325.	59.4
18	2176	2303	31.3	62.3	23030.	15.7
19	2304	2431	62.5	93.5	6.	.0
20	2432	2559	93.8	124.8	0.	0.0
21	2560	2687	125.0	156.0	2.	.0
22	2688	2815	156.3	187.3	1.	.0
23	2816	2943	187.5	218.5	0.	0.0
24	2944	3071	218.8	249.8	0.	0.0
25	3072	3199	250.0	281.0	24.	.0
26	3200	3327	281.3	312.3	2.	.0
27	3328	3455	312.5	343.5	0.	0.0
28	3456	3583	343.8	374.8	0.	0.0
29	3584	3711	375.0	406.0	1.	.0
30	3712	3839	406.3	437.3	0.	0.0
31	3840	3967	437.5	468.5	4.	.0
32	3968	4095	468.8	499.8	2.	.0

3

The output of the histogram analysis from DBHIZ is shown in Figure 10.9. The histogram is used as a quick check of the data as well as a means of determining reasonable plotting bounds for daily magnetograms.

1. This line shows the data channel and the total number of data points in the histogram.
2. MIN and MAX are the minimum and maximum number of counts for the channel being examined. FMIN and FMAX are these numbers converted to nanoteslas for channels 1, 2, and 3, and mV/km for channels 4 and 5.
3. The body of the output tells the histogram interval or bin number (BIN) and the bounds in counts (MIN and MAX) and physical units (FMIN and FMAX). The column labeled N contains the number of data points in the bin, and the last column expresses this number as a percentage of the total number of data points.

The histogram can be used to spot malfunctioning recorders. For example, if most of the data fell into one bin (particularly bins 1 or 32), it is probably a good guess that the magnetometer or electric field amplifier was saturated.

11. Appendix C - Program Listings

This section contains listings of all the programs used in transcription. They are presented in the order listed below. The list also shows the routine name, type of routine, and language used.

Page

```
FTN,L
      PROGRAM TRANZ,3,80
C
C---- PROGRAM TRANZ IS USED TO TRANSCRIBE DATA FROM SEA DATA
C     CASSETTES TO 9-TRACK MAGNETIC TAPE.  THE PROGRAM INPUTS
C     SEVERAL TRANSCRIPTION OPTIONS AND TRANSCRIPTION PARAMETERS
C     BEFORE TRANSCRIPTION BEGINS.  PROGRAM CASDS READS DATA
C     FROM THE CASSETTE READER AND WRITES THE DATA ONTO DISC.
C     CONTROL NEXT PASSES TO PROGRAM UNPKZ WHICH UNPACKS THE
C     DATA, AND MARKS AND REPORTS READER ERRORS.  THIS
C     IS FOLLOWED BY PROGRAM EDITZ WHICH ELIMINATES PARITY
C     ERRORS IF REQUESTED, CHECKS AND CORRECTS THE HEADER,
C     AND FIXES THE CLOCK VALUES IF THEY ARE NOT INCREMENTING
C     IN THE PROPER MANNER.  LASTLY PROGRAM HISTZ IS RUN. FINALLY
C     PROGRAM DBHIZ IS RUN WHICH SUMMARIZES DATA BREAKS AND
C     COMPUTES A HISTOGRAM OF THE DATA FOR USE IN DETERMINING
C     PLOTTING PROGRAM LIMITS.  CONTROL IS THEN RETURNED TO
C     PROGRAM TRANZ WHICH WRITES THE EDITTED DATA ONTO MAGNETIC
C     TAPE.
C
C     LOAD WITH SYSTEM COMMON, 'RU,LOADR,99,6,10,0,0'
C
C     WRITTEN BY D. V. FITTERMAN, U.S.G.S., MAY 1979
C     MODIFIED 28 JANUARY 1980
C
      DIMENSION ITIME(5),IBUF(1024),IPARM(5),PROG1(3),PROG2(3),
     *PROG3(3),PROG4(3),IAB(2),IDCB(144),IFILE(3),INFW(3)
      COMMON IHED(128)
      INTEGER PROG1,PROG2,PROG3,PROG4
      EQUIVALENCE (IAB,IA,AB),(IAB(2),IB),(NRATE,IHED(9)),
     *(NCHAN,IHED(10)),(NWORD,IHED(52)),(NSCAN,IHED(53))
      DATA IVER/5/,LUTTY/1/,LUPRT/6/,LUTAP/8/,LUDSK/10/,IFLGP/0/
      DATA IFILE/2HMA,2HGA,2HIN/,PROG1/2HCA,2HSD,2HS /,
     *PROG2/2HUN,2HPK,2HZ /,PROG3/2HED,2HIT,2HZ /,
     *PROG4/2HDB,2HHI,2HZ /
      DATA IBUF/1024*0/
C
C---- CHECK FOR MOUNTED SCRATCH PACK
      WRITE(LUTTY,1000) LUDSK
 1000 FORMAT(/" IS A SCRATCH PACK MOUNTED ON LU #",I2,
     *"? (YE OR NO) _")
      READ(LUTTY,1010) I
 1010 FORMAT(2A2)
      IF(I .NE. 2HYE) STOP
C
C---- PARITY ERROR RETENTION?
      WRITE(LUTTY,1020)
 1020 FORMAT(/" SAVE DATA WITH PARITY ERRORS? (YE OR NO) _")
      READ(LUTTY,1010) I
      IF(I .EQ. 2HYE) IFLGP=1
C
C---- REQUEST CURRENT DATE
      CALL EXEC(11,ITIME,IYEAR)
      WRITE(LUTTY,1040) IVER,ITIME(5),IYEAR
 1040 FORMAT(/" PROGRAM TRANZ  VERSION -",I3,"  DAY:",I3,
```

```
     *" YEAR:",I4/)
C
C---- CLEAR HEADER
   10 DO 20 I=4,128
   20 IHED(I)=0
C
C---- SET VERSION NUMBER AND TRANSCRIPTION DAY
      IHED(1)=IVER
      IHED(2)=ITIME(5)
      IHED(3)=IYEAR
C
C---- REQUEST HEADER INFORMATION
      WRITE(LUTTY,1050)
 1050 FORMAT(//" TAPE FILE #?  (<0 TO STOP) _")
      READ(LUTTY,*) IHED(4)
      IF(IHED(4) .LT. 0) GO TO 999
      WRITE(LUTTY,1060)
 1060 FORMAT(" LOCATION CODE?  (4 CHAR.) _")
      READ(LUTTY,1010) IHED(5),IHED(6)
   30 WRITE(LUTTY,1070)
 1070 FORMAT(" CASSETTE ID #?  (0-99) _")
      READ(LUTTY,*) IHED(7)
      IF(IHED(7) .GT. 99 .OR. IHED(7) .LT. 0) GO TO 30
   40 WRITE(LUTTY,1080)
 1080 FORMAT(" INSTRUMENT NUMBER?  (1-31) _")
      READ(LUTTY,*) IHED(8)
      IF(IHED(8) .LT. 1 .OR. IHED(8) .GT. 31) GO TO 40
   50 WRITE(LUTTY,1090)
 1090 FORMAT(" SCAN RATE? (0-7) _")
      READ(LUTTY,*) NRATE
      IF(NRATE .LT. 0 .OR. NRATE .GT. 7) GO TO 50
   60 WRITE(LUTTY,1100)
 1100 FORMAT(" CHANNELS/SCAN?  (1-7) _")
      READ(LUTTY,*) NCHAN
      IF(NCHAN .LT. 1 .OR. NCHAN .GT. 7) GO TO 60
C
C---- READ MAGNETOMETER GAINS FROM MAGAIN FILE
      CALL OPEN(IDCB,IER,IFILE,2)
      IF(IER .GT. 0) GO TO 70
      WRITE(LUTTY,1110) IER,IFILE
 1110 FORMAT(" IER=",I4," FILE=",3A2)
      GO TO 80
C
C---- READ MAGNETOMETER GAINS
   70 CALL READF(IDCB,IER,IHED(54),3,LEN,IHED(8)+1)
      CALL CLOSE(IDCB)
C
C---- ALLOW CHANGES IF DESIRED
   80 WRITE(LUTTY,1120) (IHED(I),I=54,56)
 1120 FORMAT(" MAGNETOMETER GAIN IN GAMMAS/2048 COUNTS"/
     *" HX=",I5," HY=",I5," HZ=",I5/
     *" IF OK TYPE 0'S, IF NOT TYPE NEW VALUES"/" HX   HY   HZ")
      READ(LUTTY,*) (INEW(I),I=1,3)
      DO 90 I=1,3
      IF(INEW(I) .EQ. 0) GO TO 90
```

```
            IHED(I+53)=INEW(I)
       90 CONTINUE
C
C---- CHECK FOR 3 OR LESS CHANNELS
            IF(NCHAN .LE. 3) GO TO 120
C
C---- TELLURIC LINE GAINS AND LINE LENGTHS
            WRITE(LUTTY,1130)
     1130 FORMAT(" EX GAIN? (>0, NORTH(+)  <0, SOUTH(+)) _")
            READ(LUTTY,*) IHED(57)
      100 WRITE(LUTTY,1140)
     1140 FORMAT(" EX LENGTH? (METERS) _")
            READ(LUTTY,*) IHED(59)
            IF(IHED(59) .LT. 0) GO TO 100
            WRITE(LUTTY,1150)
     1150 FORMAT(" EY GAIN? (>0, EAST(+)  <0, WEST(+)) _")
            READ(LUTTY,*) IHED(58)
      110 WRITE(LUTTY,1160)
     1160 FORMAT(" EY LENGTH? (METERS) _")
            READ(LUTTY,*) IHED(60)
            IF(IHED(60) .LT. 0) GO TO 110
C
C---- RESET TIME
      120 WRITE(LUTTY,1170)
     1170 FORMAT(" RESET TIME?  (HR MIN DAY MON YR) _")
            READ(LUTTY,*) (IHED(I),I=11,15)
            CALL CHECK(11,IHED,IER)
            IF(IER .NE. 0) GO TO 120
C
C---- OFF TIME
      130 WRITE(LUTTY,1180)
     1180 FORMAT(" OFF TIME?     (HR MIN DAY MON YR) _")
            READ(LUTTY,*) (IHED(I),I=16,20)
            CALL CHECK(16,IHED,IER)
            IF(IER .NE. 0) GO TO 130
C
C---- STOP WATCH
      140 WRITE(LUTTY,1190)
     1190 FORMAT(" STOP WATCH?  (MIN SEC TENTHS) _")
            READ(LUTTY,*) (IHED(I),I=21,23)
            IF(IHED(21) .LT. 0 .OR. IHED(22) .LT. 0 .OR.
          * IHED(22) .GE. 60 .OR. IHED(23) .LT. 0 .OR.
          * IHED(23) .GE. 10) GO TO 140
C
C---- COMMENTS
            WRITE(LUTTY,1200)
     1200 FORMAT(" COMMENTS?  (50 CHARACTERS)")
            READ(LUTTY,1210) (IHED(I),I=27,51)
     1210 FORMAT(25A2)
C
C---- COMPUTE NSCAN AND NWORD
            NSCAN=24/NCHAN
            NWORD=NCHAN*NSCAN+8
            IHED(25)=32
C
```

```
C---- 1, 2, 3, 4, OR 6 CHANNELS
      IHED(24)=30
      IHED(26)=960
      ICHAR=86
      IF(NCHAN .LE. 4 .OR. NCHAN .EQ. 6) GO TO 150
C
C---- 5 CHANNELS
      IHED(24)=26
      IHED(26)=832
      ICHAR=74
      IF(NCHAN .EQ. 5) GO TO 150
C
C---- 7 CHANNELS
      IHED(24)=27
      IHED(26)=864
      ICHAR=77
  150 WRITE(LUTTY,1220) ICHAR
 1220 FORMAT(" SET READER CHARACTERS/RECORD=",I2,".  PRESS READ.")
C
C---- CHECK THAT TTY IS AVAILABLE
  160 CALL EXEC(13,LUTTY,IEQT5)
      IF(IAND(IEQT5,140000B) .NE. 0) GO TO 160
C
C---- SCHEDULE CASSETTE TO DISC WITH WAIT
      CALL EXEC(9,PROG1,IHED(26))
C
C---- GET PARAMETERS FROM CASDS
      CALL RMPAR(IPARM)
C
C---- CHECK FOR DMA'S NOT AVAILABLE AND RESCHEDULE CASDS
      IF(IPARM(4) .NE. 1B) GO TO 170
      WRITE(LUTTY,1230)
 1230 FORMAT(" BOTH DMA'S NOT AVAILABLE, RESCHEDULE CASDS")
      GO TO 160
  170 WRITE(LUTTY,1240) IPARM
 1240 FORMAT(/" CASDS COMPLETED"/" CASRC=",I6," DISRC=",I6,
     *" BADRC=",I6," COMST=",@6,"B LSTRK=",I6//)
C
C---- WRITE TRANSCRIPTION REPORT HEADER
      CALL EXEC(3,1100B+LUPRT,10)
      WRITE(LUPRT,1250) (IHED(I),I=1,23),(IHED(I),I=27,51)
 1250 FORMAT(" VER=",I3," TRANSCRIPTION DAY=",I3," YEAR=",I4/
     *" TAPE FILE #",I4," LOC=",2A2," CASS ID #",I2," INST. #",I2/
     *" SCAN RATE=",I1," CHAN/SCAN=",I1/
     *" RESET TIME=",I2,":",I2," DAY=",I2," MON=",I2," YR=",I4/
     *" OFF   TIME=",I2,":",I2," DAY=",I2," MON=",I2," YR=",I4,
     *" SW=",I2,":",I2,".",I1/1X,25A2)
      WRITE(LUPRT,1240) IPARM
C
C---- SCHEDULE UNPKZ WITH WAIT
      CALL EXEC(9,PROG2,IPARM(2),IHED(26),IHED(24),IHED(52))
C
C---- PICK UP NUMBER OF DISC RECORDS AND VERSION NUMBER
      CALL RMPAR(IPARM)
C
```

```
C---- UPDATE VERSION NUMBER=10*TRANZ+UNPKZ
      IHED(1)=10*IHED(1)+IPARM(2)
C
C---- CHANGE HEADER NUMBER OF WORDS PER DISC/TAPE RECORD
      IHED(26)=1024
      IF(NCHAN .EQ. 7) IHED(26)=928
      IF(NCHAN .EQ. 5) IHED(26)=896
C
C---- SCHEDULE EDITZ WITH WAIT
      CALL EXEC(9,PROG3,IPARM(1),IFLGP)
C
C---- PICK UP DISC RECORDS, COMPLETION CODE, VERSION NUMBER
      CALL RMPAR(IPARM)
C
C---- CHECK FOR UNSATISFACTORY DATA EDITTING
      IF(IPARM(2) .EQ. 0) GO TO 180
      WRITE(LUTTY,1260)
 1260 FORMAT(/" TRANSCRIPTION TERMINATED BY EDITZ - NO TAPE WRITTEN"/)
      GO TO 10
C
C---- UPDATE VERSION NUMBER=100*TRANZ+10*UNPKZ+EDITZ
  180 IHED(1)=10*IHED(1)+IPARM(3)
C
C---- SCHEDULE DBHIZ WITH WAIT
      CALL EXEC(9,PROG4,IPARM(1))
C
C---- PICK UP NUMBER OF DISC RECORDS AND VERSION NUMBER
      CALL RMPAR(IPARM)
C
C---- UPDATE VERSION NUMBER=1000*TRANZ+100*UNPKZ+10*EDITZ+DBHIZ
      IHED(1)=10*IHED(1)+IPARM(2)
C
C---- OUTPUT HEADER
  190 AB=EXEC(2,100B+LUTAP,IHED,128)
C
C---- TEST FOR END OF TAPE
      IF(IAND(IA,40B) .NE. 0) GO TO 200
      ITRK=0
      ISEC=-16
      DO 210 I=1,IPARM
      IF(ISEC .EQ. 80) ITRK=ITRK+1
      ISEC=MOD(ISEC+16,96)
C
C---- READ DATA FROM DISC
      CALL EXEC(1,100B+LUDSK,IBUF,1024,ITRK,ISEC)
C
C---- WRITE DATA ONTO TAPE
      AB=EXEC(2,100B+LUTAP,IBUF,IHED(26))
C
C---- TEST FOR END OF TAPE
      IF(IAND(IA,40B) .EQ. 0) GO TO 210
  200 WRITE(LUTTY,1270)
 1270 FORMAT(" END OF TAPE"/" MOUNT NEW TAPE AND TYPE 'GO,TRANS'")
C
C---- BACKSPACE FILE
```

```
            CALL EXEC(3,1400B+LUTAP)
      C
      C---- SKIP FORWARD OVER EOF
            CALL EXEC(3,300B+LUTAP)
      C
      C---- WRITE SECOND EOF
            CALL EXEC(3,100B+LUTAP)
            PAUSE
            GO TO 190
        210 CONTINUE
      C
      C---- WRITE END OF FILE
            CALL EXEC(3,100B+LUTAP)
            WRITE(LUTTY,1280) IHED(1),IPARM(1)
       1280 FORMAT(" VERSION=",I4,2X,I4," RECORDS WRITTEN")
            GO TO 10
      C
      C---- WRITE SECOND EOF
        999 CALL EXEC(3,100B+LUTAP)
            STOP
            END
```

FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 03075 COMMON = 00128

```
      SUBROUTINE CHECK(I,IHED,IER)
C
C---- SUBROUTINE TO CHECK THAT A GIVEN HOUR, MINUTE, DAY, MONTH,
C     YEAR COMBINATION IS A REALIZABLE TIME.
      DIMENSION IDAYS(12),IHED(128)
      DATA IDAYS/31,28,31,30,31,30,31,31,30,31,30,31/
      IER=0
C
C---- CHECK THE HOURS
      IF(IHED(I) .GE. 24 .OR. IHED(I) .LT. 0) IER=1
C
C---- CHECK THE MINUTES
      IF(IHED(I+1) .GE. 60 .OR. IHED(I+1) .LT. 0) IER=1
C
C---- SET UP FOR POSSIBLE LEAP YEAR
      J=0
      IF(IHED(I+3) .EQ. 2) J=LEAP(IHED(I+4))
C
C---- CHECK THE DAY
      IF(IHED(I+2) .GT. IDAYS(IHED(I+3))+J .OR.
     * IHED(I+2) .LE. 0) IER=1
C
C---- CHECK THE MONTH
      IF(IHED(I+3) .GT. 12 .OR. IHED(I+3) .LE. 0) IER=1
C
C---- CHECK THE YEAR
      IF(IHED(I+4) .LT. 1970) IER=1
      RETURN
      END


 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00168      COMMON = 00000
```

```
      INTEGER FUNCTION LEAP(IYEAR)
C
C---- DETERMINES IF IYEAR IS A LEAP YEAR
C     RETURNS: LEAP=1 FOR A YEAR YEAR
C              LEAP=0 FOR ANY OTHER YEAR
      LEAP=0
      IF(MOD(IYEAR,4) .NE. 0) GO TO 20
      IF(MOD(IYEAR,100) .NE. 0) GO TO 10
      IF(MOD(IYEAR,400) .NE. 0) GO TO 20
   10 LEAP=1
   20 RETURN
      END
```

FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 00046 COMMON = 00000

```
0338        END$
```

```
0001                    ASMB,L,T,C
CASDS R 000000
$LIBR X 000001
$LIBX X 000002
RMPAR X 000003
PRTN  X 000004
EXEC  X 000005
LOOP  R 000041
SCAN1 R 000043
TEST1 R 000052
SCAN2 R 000101
TEST2 R 000110
ERROR R 000136
ABORT R 000141
SETA  R 000153
SETB  R 000165
WRITA R 000177
WRITB R 000227
SEEK  R 000257
STATS R 000301
CHECK R 000316
STEOF R 000332
STF   R 000337
OPSTP R 000341
FLUSH R 000345
SIX   R 000363
BUMP  R 000376
DISPL R 000417
INSCT R 000437
DSFUL R 000442
DISCE R 000445
EXIT  R 000450
RETRN R 000455
A       000000
B       000001
DC      000011
CC      000012
CASS    000021
ARTRN R 000463
CW2AO R 000464
CW2BO R 000465
CW2AI R 000466
CW2BI R 000467
COD6  R 000470
CW1IN R 000471
COUNT R 000472
CW1OT R 000473
INCMD R 000474
SKCMD R 000475
STCMD R 000476
WDDS  R 000477
ILENG R 000504
SYSWR R 000505
BUFA  R 000506
BUFB  R 002506
HEAD  R 004506
```

```
CASRC R 004507
DISRC R 004510
BADRC R 004511
COMST R 004512
TRACK R 004513
SECTR R 004514
FOF   R 004515
TEMP  R 004516
**  NO ERRORS PASS#1 **RTE ASMB 760924**
```

```
                        ASMB,L,T,C
      00000                   NAM CASDS,3,80
                              ENT CASDS
                              EXT $LIBR,$LIBX,RMPAR,PRTN,EXEC
*
* CASSETTE TO DISC TRANSFER ROUTINE.  TRANSFERS
* DATA FROM CASSETTE TO DISC VIA INTERMEDIATE
* BUFFERS.  PING-PONGS BETWEEN THE BUFFERS FILLING
* ONE AS THE OTHER IS BEING EMPTIED.
* VIA RMPAR THE PROGRAM PICKS UP THE NUMBER OF
* CASSETTE WORDS TO BE STORED IN A DISC RECORD
* OF LENGTH 1024.
*
* RETURNS VIA PRTN THE FOLLOWING PARAMETERS
*
*   CASRC - # OF CASSETTE RECORDS READ
*   DISRC - # OF DISC RECORDS WRITTEN
*   BADRC - # OF BAD CASSETTE RECORDS
*   COMST - COMPLETION STATUS
*         = 000000B SATISFACTORY COMPLETION
*         = 000001B BOTH DMA'S NOT AVAILABLE
*         = 177777B DISC FULL
*         = XXXXXXB AND HLT 77B DISC ERROR
*           "S" AND COMST SET TO DISC STATUS WORD
*   TRACK = CURRENT TRACK ADDRESS TO BE WRITTEN
*
* WRITTEN BY D. V. FITTERMAN
*           BRANCH OF ELETROMAGNETISM AND GEOMAGNETISM
*           U. S. GEOLOGICAL SURVEY
*           DENVER, COLORADO  80225
*
*           MODIFIED 2 OCTOBER 1979
*
      00000 000000  CASDS NOP
      00001 016003X       JSB RMPAR    GET # OF CASSETTE WORDS
      00002 000004R       DEF *+2      PER DISC RECORD OF 1024
      00003 000477R       DEF WDDS
      00004 016001X       JSB $LIBR    TURN OFF INTERRUPTS
      00005 000000        NOP          AND MEMORY PROTECT
      00006 065654        LDB 1654B    LOAD 1ST AND 2ND WORD
      00007 104200        DLD B,I      OF INTERRUPT TABLE
      00010 100001
      00011 002002        SZA          1ST DMA AVAILABLE?
      00012 026136R       JMP ERROR    NO, ERROR!
      00013 006002        SZB          YES, 2ND DMA AVAILABLE?
      00014 026136R       JMP ERROR    NO, ERROR!
      00015 102501        LIA 1B       YES, SAVE SYSTEM SWITCH
      00016 072505R       STA SYSWR    REGISTER
      00017 062513R       LDA TRACK
      00020 102601        OTA 1B       OUTPUT CURRENT TRACK
      00021 062477R       LDA WDDS
      00022 003004        CMA,INA      COMPUTE LENGTH OF INPUT BUFFERS
      00023 072504R       STA ILENG
      00024 062466R       LDA CW2AI
      00025 032517R       IOR =B100000 FORM INPUT STARTING ADDRESSES
      00026 072466R       STA CW2AI
```

```
00027 062467R       LDA CW2BI
00030 032517R       IOR =B100000
00031 072467R       STA CW2BI
00032 016141R       JSB ABORT    TURN OFF DMA, DISC, CASS, AND INTERRUPT
00033 016153R       JSB SETA     SET UP DMA INPUT TO BUFA
00034 103721        STC CASS,C   START FILLING BUFA
00035 103706        STC 6B,C     START DMA TRANSFER
00036 105755        LDY =B6      SET "Y" TO DMA S.C.
00037 004520R
00040 016165R       JSB SETB     SET UP DMA INPUT FOR BUFB
00041 105745  LOOP  LDX =B0      ZERO "X"
00042 004521R
00043 066504R SCAN1 LDB ILENG    LOAD PAST DMA WORD COUNT
00044 102501        LIA 1B       TEST FOR OPERATOR STOP
00045 002020        SSA          BIT 15 = ZERO?
00046 026052R       JMP TEST1    NO, EXIT
00047 102502        LIA 2B       YES, LOAD PRESENT DMA WORD COUNT REGIST
00050 050001        CPA B        PAST = PRESENT?
00051 026044R       JMP SCAN1+1  YES, WAIT FOR SOME DATA TO BE READ
00052 101742  TEST1 LAX BUFA     LOAD DATA
00053 000506R
00054 016316R       JSB CHECK              AND TEST
00055 105760        ISX          INCREMENT "X"
00056 101744        CXA          "X" TO "A"
00057 052477R       CPA WDDS     "X" = WDDS?
00060 026063R       JMP *+3      YES, DONE SCANNING DATA
00061 006004        INB          NO, INCREMENT "B"
00062 026044R       JMP SCAN1+1    AND SCAN SOME MORE
00063 102306        SFS 6B       COMPLETION FLAG SET?
00064 026063R       JMP *-1      NO, WAIT
00065 103721        STC CASS,C   YES, START FILLING BUFB
00066 103707        STC 7B,C     START DMA TRANSFER
00067 105755        LDY =B7      SET "Y" TO DMA S.C.
00070 004522R
00071 016177R       JSB WRITA    OUTPUT BUFA
00072 016153R       JSB SETA     SET UP DMA INPUT FOR BUFA
00073 016376R       JSB BUMP     INCREMENT HEAD/SECTOR/TRACK ADDRESS
00074 062512R       LDA COMST
00075 002002        SZA          BIT   = ZERO?
00076 026450R       JMP EXIT     NO, DISC FULL, EXIT
00077 105745        LDX =B0      ZERO "X"
00100 004521R
00101 066504R SCAN2 LDB ILENG    LOAD PAST DMA WORD COUNT
00102 102501        LIA 1B       TEST FOR OPERATOR STOP
00103 002020        SSA          BIT 15 = ZERO?
00104 026110R       JMP TEST2    NO, EXIT
00105 102503        LIA 3B       YES, LOAD PRESENT DMA WORD COUNT REGIST
00106 050001        CPA B        PAST = PRESENT?
00107 026102R       JMP SCAN2+1  YES, WAIT FOR SOME DATA TO BE READ
00110 101742  TEST2 LAX BUFB     LOAD DATA
00111 002506R
00112 016316R       JSB CHECK              AND TEST
00113 105760        ISX          INCREMENT "X"
00114 101744        CXA          "X" TO "A"
00115 052477R       CPA WDDS     "X" = WDDS?
00116 026121R       JMP *+3      YES, DONE SCANNING DATA
```

```
00117 006004          INB              NO, INCREMENT "B"
00120 026102R         JMP SCAN2+1          AND SCAN SOME MORE
00121 102307          SFS 7B           COMPLETION FLAG SET?
00122 026121R         JMP *-1          NO, WAIT
00123 103721          STC CASS,C       YES, START FILLING BUFA
00124 103706          STC 6B,C         START DMA TRANSFER
00125 105755          LDY =B6          SET "Y" TO DMA S.C.
00126 004520R
00127 016227R         JSB WRITB        OUTPUT BUFB
00130 016165R         JSB SETB         SET UP DMA INPUT FOR BUFB
00131 016376R         JSB BUMP         INCREMENT HEAD/SECTOR/TRACK ADDRESS
00132 062512R         LDA COMST
00133 002002          SZA              BIT 0 = ZERO?
00134 026450R         JMP EXIT         NO, DISC FULL, EXIT
00135 026041R         JMP LOOP
00136 002404   ERROR  CLA,INA          ERROR FOR BOTH DMA'S
00137 072512R         STA COMST        NOT AVAILABLE
00140 026453R         JMP EXIT+3
00141 000000   ABORT  NOP              SHUT OFF DISC, CASS, DMA'S
00142 103100          CLF 0            AND INTERRUPTS
00143 102106          STF 6B
00144 102107          STF 7B
00145 106721          CLC CASS
00146 106711          CLC DC
00147 106712          CLC CC
00150 106706          CLC 6B           PREVENT ILLEGAL INTERRUPT FROM 6B
00151 106707          CLC 7B           PREVENT ILLEGAL INTERRUPT FROM 7B
00152 126141R         JMP ABORT,I
00153 000000   SETA   NOP              SET UP BUFA DMA FOR INPUT
00154 062471R         LDA CW1IN        STC, NO CLC, S.C.
00155 102606          OTA 6B           OUTPUT 1ST DMA CONTROL WORD
00156 106702          CLC 2B           PREPARE FOR 2ND DMA CONTROL WORD
00157 062466R         LDA CW2AI
00160 102602          OTA 2B           OUTPUT 2ND DMA CONTROL WORD
00161 102702          STC 2B           PREPARE FOR 3RD DMA CONTROL WORD
00162 062504R         LDA ILENG
00163 102602          OTA 2B           OUTPUT 3RD DMA CONTROL WORD
00164 126153R         JMP SETA,I
00165 000000   SETB   NOP              SET UP BUFB DMA FOR INPUT
00166 062471R         LDA CW1IN        STC, NO CLC, S.C.
00167 102607          OTA 7B           OUTPUT 1ST DMA CONTROL WORD
00170 106703          CLC 3B           PREPARE FOR 2ND DMA CONTROL WORD
00171 062467R         LDA CW2BI
00172 102603          OTA 3B           OUTPUT 2ND DMA CONTROL WORD
00173 102703          STC 3B           PREPARE FOR 3RD DMA CONTROL WORD
00174 062504R         LDA ILENG
00175 102603          OTA 3B           OUTPUT 3RD DMA CONTROL WORD
00176 126165R         JMP SETB,I
00177 000000   WRITA  NOP              WRITE BUFA TO DISC
00200 016257R         JSB SEEK         MOVE TO DESIRED TRACK
00201 106711          CLC DC           INSURE DATA AND COMMAND
00202 106712          CLC CC           CHANNELS ARE RESET
00203 062473R         LDA CW1OT        STC, CLC, S.C.
00204 102606          OTA 6B           OUTPUT 1ST DMA CONTROL WORD
00205 106702          CLC 2B           PREPARE FOR 2ND DMA CONTROL WORD
00206 062464R         LDA CW2AO
```

```
00207 102602         OTA 2B        OUTPUT 2ND DMA CONTROL WORD
00210 102702         STC 2B        PREPARE FOR 3RD DMA CONTROL WORD
00211 062472R        LDA COUNT
00212 102602         OTA 2B        OUTPUT 3RD DMA CONTROL WORD
00213 102711         STC DC
00214 102111         STF DC        ENABLE DMA TO DATA CHANNEL TRANSFER
00215 103706         STC 6B,C      ACTIVATE DMA
00216 062474R        LDA INCMD     LOAD INITIALIZE COMMAND WORD
00217 102612         OTA CC        OUTPUT COMMAND WORD TO COMMAND CHANNEL
00220 103712         STC CC,C      INITIATE WRITE COMMAND
00221 102312         SFS CC        IS WRITE COMPLETE?
00222 026221R        JMP *-1       NO,WAIT
00223 102106         STF 6B        YES, ABORT DMA JUST IN CASE IT'S HUNG
00224 016301R        JSB STATS     CHECK STATUS
00225 036510R        ISZ DISRC     INCREMENT DISC RECORDS
00226 126177R        JMP WRITA,I
00227 000000  WRITB  NOP           WRITE BUFB TO DISC
00230 016257R        JSB SEEK      MOVE TO DESIRED TRACK
00231 106711         CLC DC        INSURE DATA AND COMMAND
00232 106712         CLC CC        CHANNELS ARE RESET
00233 062473R        LDA CW1OT     STC, CLC, S.C.
00234 102607         OTA 7B        OUTPUT 1ST DMA CONTROL WORD
00235 106703         CLC 3B        PREPARE FOR 2ND DMA CONTROL WORD
00236 062465R        LDA CW2BO
00237 102603         OTA 3B        OUTPUT 2ND DMA CONTROL WORD
00240 102703         STC 3B        PREPARE FOR 3RD DMA CONTROL WORD
00241 062472R        LDA COUNT
00242 102603         OTA 3B        OUTPUT 3RD DMA CONTROL WORD
00243 102711         STC DC
00244 102111         STF DC        ENABLE DMA TO DATA CHANNEL TRANSFER
00245 103707         STC 7B,C      ACTIVATE DMA
00246 062474R        LDA INCMD     LOAD INITIALIZE COMMAND WORD
00247 102612         OTA CC        OUTPUT COMMAND WORD TO COMMAND CHANNEL
00250 103712         STC CC,C      INITIATE WRITE COMMAND
00251 102312         SFS CC        IS WRITE COMPLETE?
00252 026251R        JMP *-1       NO, WAIT
00253 102107         STF 7B        YES, ABORT DMA JUST IN CASE IT'S HUNG
00254 016301R        JSB STATS     CHECK STATUS
00255 036510R        ISZ DISRC     INCREMENT DISC RECORDS
00256 126227R        JMP WRITB,I
00257 000000  SEEK   NOP           SEEK A HEAD/TRACK/SECTOR
00260 106711         CLC DC        INSURE DATA AND COMMAND
00261 106712         CLC CC        CHANNELS ARE RESET
00262 062513R        LDA TRACK
00263 102611         OTA DC        OUTPUT TRACK ADDRESS TO DATA CHANNEL
00264 103711         STC DC,C
00265 062475R        LDA SKCMD
00266 102612         OTA CC        OUTPUT SEEK COMMAND TO COMMAND CHANNEL
00267 103712         STC CC,C      INITIATE SEEK COMMAND
00270 102311         SFS DC        FIRST ADDRESS WORD ACCEPTED?
00271 026270R        JMP *-1       NO, WAIT
00272 062506R        LDA HEAD      YES, FORM HEAD/SECTOR ADDRESS
00273 032514R        IOR SECTR
00274 102611         OTA DC        OUTPUT HEAD/SECTOR ADDRESS TO DC
00275 103711         STC DC,C      INITIATE SEEK
00276 102312         SFS CC        SEEK OPERATION COMPLETE?
```

```
PAGE 0007 #01
00277 026276R        JMP *-1        NO, WAIT
00300 126257R        JMP SEEK,I
00301 000000  STATS  NOP            GET DISC STATUS WORD
00302 106711         CLC DC         INSURE DATA AND COMMAND
00303 106712         CLC CC         CHANNELS ARE RESET
00304 103711         STC DC,C       READY DATA CHANNEL TO SEND STATUS WORD
00305 062476R        LDA STCMD
00306 102612         OTA CC         OUTPUT STATUS COMMAND
00307 103712         STC CC,C       INITIATE STATUS CHECK COMMAND
00310 102311         SFS DC         STATUS WORD READY?
00311 026310R        JMP *-1        NO, WAIT
00312 102511         LIA DC         YES, LOAD STATUS WORD
00313 000010         SLA            ANY ERROR BIT = ONE?
00314 026445R        JMP DISCE      YES, GO TO DISC ERROR ROUTINE
00315 126301R        JMP STATS,I    NO, RETURN
00316 000000  CHECK  NOP            CHECK FOR ERRORS, EOR, EOF, & OP STOPS
00317 002021         SSA,RSS        BIT 15 = ONE?
00320 026341R        JMP OPSTP      NO, DATA WORD, BUT NOT LAST
00321 000010         SLA            YES, BIT 0 = ZERO?
00322 026325R        JMP *+3        NO, MESSAGE WORD
00323 036507R        ISZ CASRC      YES, PROCESS LAST DATA WORD IN RECORD
00324 026341R        JMP OPSTP
00325 001226         RAL,ELA        PUT BIT 14 IN "E"
00326 012523R        AND =B017000
00327 002002         SZA            "A" = ZERO?
00330 036511R        ISZ BADRC      NO, ERROR IN RECORD
*
* REMOVE UNTIL EOF ELECTRONICS IS WORKING
*      SEZ,RSS       YES, TEST FOR EOF  "E" = ONE?.
00331 026341R        JMP OPSTP      NO
00332 003400  STEOF  CCA            YES, SET EOF FLAG
00333 072515R        STA EOF
00334 101754         CYA            STOP INPUT DMA
00335 032524R        IOR =B102100   FORM STF COMMAND
00336 072337R        STA STF
00337 000000  STF    NOP            SET FLAG OF INPUT DMA
00340 026345R        JMP FLUSH      FLUSH OUT BUFFER
00341 102501  OPSTP  LIA 1B
00342 002020         SSA            BIT 15 = ZERO?
00343 026332R        JMP STEOF      NO, OPERATOR TERMINATE!!!
00344 126316R        JMP CHECK,I    YES, KEEP PROCESSING DATA
00345 101754  FLUSH  CYA            FLUSH OUT LAST BUFFER
00346 052520R        CPA =B6        "A" = 6B?
00347 026363R        JMP SIX        YES
00350 102503         LIA 3B         NO, ZERO END OF BUFB
00351 072516R        STA TEMP
00352 042477R        ADA WDDS
00353 006400         CLB
00354 042465R        ADA CW2BO      FORM STARTING ADDRESS
00355 174000         STB A,I
00356 002004         INA
00357 036516R        ISZ TEMP
00360 026355R        JMP *-3
00361 016227R        JSB WRITB      FLUSH BUFB
00362 026450R        JMP EXIT
00363 102502  SIX    LIA 2B         ZERO END OF BUFA
```

```
00364 072516R        STA TEMP
00365 042477R        ADA WDDS
00366 006400         CLB
00367 042464R        ADA CW2AO    FORM STARTING ADDRESS
00370 174000         STB A,I
00371 002004         INA
00372 036516R        ISZ TEMP
00373 026370R        JMP *-3
00374 016177R        JSB WRITA        FLUSH BUFA
00375 026450R        JMP EXIT     RESTORE SYSTEM
00376 000000  BUMP   NOP          INCREMENT HEAD/TRACK/SECTOR
00377 062514R        LDA SECTR
00400 052525R        CPA =D16     SECTR = 16?
00401 026403R        JMP *+2      YES
00402 026437R        JMP INSCT    NO, ADD 8 TO SECTOR ADDRESS
00403 002400         CLA
00404 072514R        STA SECTR    CLEAR SECTOR ADDRESS
00405 062506R        LDA HEAD     TOGGLE HEAD ADDRESS
00406 042526R        ADA =B400
00407 012527R        AND =B777
00410 072506R        STA HEAD
00411 002002         SZA          HEAD = ZERO?
00412 026417R        JMP DISPL    NO, CHANGE DISPLAY
00413 062513R        LDA TRACK    YES, CHECK TRACK
00414 052530R        CPA =D200    TRACK = 200?
00415 026442R        JMP DSFUL    YES, DISC FULL!!!
00416 036513R        ISZ TRACK    NO, INCREMENT TRACK
00417 102501  DISPL  LIA 1B       OUTPUT TRACK AND
00420 001600         ELA          SECTOR ADDRESSES
00421 062513R        LDA TRACK    BEING CAREFUL TO
00422 001727         ALF,ALF      PRESERVE BIT 15
00423 066514R        LDB SECTR
00424 005020         BLS,BLS
00425 030001         IOR B
00426 001323         RAR,RAR
00427 001323         RAR,RAR
00430 001500         ERA
00431 066506R        LDB HEAD
00432 005727         BLF,BLF
00433 005222         RBL,RBL
00434 030001         IOR B
00435 102601         OTA 1B
00436 126376R        JMP BUMP,I
00437 042531R INSCT  ADA =D8      ADD 8 TO SECTOR ADDRESS
00440 072514R        STA SECTR
00441 026417R        JMP DISPL
00442 003400  DSFUL  CCA
00443 072512R        STA COMST    SET FULL DISC FLAG
00444 126376R        JMP BUMP,I
00445 102601  DISCE  OTA 1B       SET DISPLAY REGISTER AND COMST
00446 072512R        STA COMST    TO DISC STATUS WORD
00447 102077         HLT 77B
00450 062505R EXIT   LDA SYSWR    RESTORE SYSTEM
00451 102601         OTA 1B       SWITCH REGISTER
00452 016141R        JSB ABORT    TURN OFF DMA'S, DISC, CASS
00453 016002X        JSB $LIBX    TURN ON INTERRUPTS AND
```

```
PAGE 0009 #01
0329  00454 000463R         DEF ARTRN    MEMORY PROTECT
0330  00455 016004X RETRN JSB PRTN     RETURN PARAMETERS TO FATHER
0331  00456 000460R         DEF *+2
0332  00457 004507R         DEF CASRC
0333  00460 016005X         JSB EXEC     TERMINATE PROGRAM
0334  00461 000463R         DEF *+2
0335  00462 000470R         DEF COD6
0336*
0337* DEVICES, CONTROL WORDS, AND COMMANDS
0338  00000         A     EQU 0B
0339  00001         B     EQU 1B
0340  00011         DC    EQU 11B      DISC DATA CHANNEL
0341  00012         CC    EQU 12B      DISC COMMAND CHANNEL
0342  00021         CASS  EQU 21B      CASSETTE SELECT CODE
0343  00463 000455R ARTRN DEF RETRN
0344  00464 000506R CW2AO DEF BUFA
0345  00465 002506R CW2BO DEF BUFB
0346  00466 000506R CW2AI DEF BUFA
0347  00467 002506R CW2BI DEF BUFB
0348  00470 000006  COD6  DEC 6
0349  00471 100021  CW1IN OCT 100021   STC, NO CLC, S.C. = 21B
0350  00472 176000  COUNT OCT 176000   -1024
0351  00473 120011  CW1OT OCT 120011   STC, CLF, S.C. = 11B FOR HP7900
0352  00474 010000  INCMD OCT 010000   INITIALIZE COMMAND WORD
0353  00475 030000  SKCMD OCT 030000   SEEK       COMMAND WORD
0354  00476 000000  STCMD OCT 000000   STATUS     COMMAND WORD
0355  00477 000000  WDDS  BSS 5        # OF CASSETTE RECORDS/DISC BUFFER
0356  00504 000000  ILENG BSS 1
0357  00505 000000  SYSWR BSS 1        SYSTEM SWITCH REGISTER
0358  00506 000000  BUFA  BSS 1024
0359  02506 000000  BUFB  BSS 1024
0360  04506 000000  HEAD  OCT 0        CURRENT HEAD
0361  04507 000000  CASRC BSS 1        # OF CASSETTE RECORDS READ
0362  04510 000000  DISRC BSS 1        # OF DISC RECORDS
0363  04511 000000  BADRC BSS 1        # OF BAD CASSETTE RECORDS
0364  04512 000000  COMST OCT 0        COMPLETION STATUS
0365  04513 000000  TRACK OCT 0        CURRENT TRACK
0366  04514 000000  SECTR OCT 0        CURRENT SECTOR
0367  04515 000000  EOF   OCT 0        END OF FILE FLAG
0368  04516 000000  TEMP  BSS 1        COUNT FOR FLUSHING FINAL BUFFER
      04517 100000
      04520 000006
      04521 000000
      04522 000007
      04523 017000
      04524 102100
      04525 000020
      04526 000400
      04527 000777
      04530 000310
      04531 000010
0369                      END CASDS
** NO ERRORS *TOTAL **RTE ASMB 760924**
```

CROSS-REFERENCE SYMBOL TABLE

$LIBR	00004	00038					
$LIBX	00004	00328					
=B0	.....	00065	00092				
=B01700	....	00240					
=B10000	....	00054	00057				
=B10210	....	00250					
=B400	.....	00291					
=B6	.....	00063	00112	00259	N		
=B7	.....	00085					
=B777	.....	00292					
=D16	.....	00285					
=D200	.....	00297					
=D8	.....	00316					
A	00338	00266	00277				
ABORT	00123	00059	00132	00327	N		
ARTRN	00343	00329					
B	00339	00041	00071	00098	00306	00313	O
BADRC	00363	00242					
BUFA	00358	00073	00344	00346			
BUFB	00359	00100	00345	00347			
BUMP	00283	00088	00115	00315	00321		
CASDS	00034	00003	00369				
CASRC	00361	00237	00332				
CASS	00342	00061	00083	00110	00127		
CC	00341	00129	00156	00169	00170	00171	00180
	00193	00194	00195	00203	00208	00209	00216
	00221	00224	00225				
CHECK	00232	00074	00101	00257			

CROSS-REFERENCE SYMBOL TABLE

COD6	00348	00335					
COMST	00364	00089	00116	00121	00320	00323	
COUNT	00350	00163	00187				
CW1IN	00349	00134	00144				
CW1OT	00351	00157	00181				
CW2AI	00346	00053	00055	00137			
CW2AO	00344	00160	00276				
CW2BI	00347	00056	00058	00147			
CW2BO	00345	00184	00265				
DC	00340	00128	00155	00165	00166	00179	00189
	00190	00202	00205	00206	00210	00214	00215
	00220	00222	00226	00228			
DISCE	00322	00230					
DISPL	00300	00295	00318				
DISRC	00362	00175	00199				
DSFUL	00319	00298					
EOF	00367	00248					
ERROR	00120	00043	00045				
EXEC	00004	00333					
EXIT	00325	00091	00118	00122	00271	00282	
FLUSH	00258	00253					
HEAD	00360	00212	00290	00293	00310		
ILENG	00356	00052	00066	00093	00140	00150	
INCMD	00352	00168	00192				
INSCT	00316	00287					
LOOP	00065	00119					
OPSTP	00254	00234	00238	00246			
PRTN	00004	00330					
RETRN	00330	00343					

CASDS
CROSS-REFERENCE SYMBOL TABLE

RMPAR	00004	00035					
SCAN1	00066	00072	00080				
SCAN2	00093	00099	00107				
SECTR	00366	00213	00284	00289	00304	00317	
SEEK	00201	00154	00178	00218			
SETA	00133	00060	00087	00142			
SETB	00143	00064	00114	00152			
STX	00272	00260					
SKCMD	00353	00207					
STATS	00219	00174	00198	00231			
STCMD	00354	00223					
STEOF	00247	00256					
STF	00252	00251					
SYSWR	00357	00047	00325				
TEMP	00368	00262	00268	00273	00279		
TEST1	00073	00069					
TEST2	00100	00096					
TRACK	00365	00048	00204	00296	00299	00302	
WDDS	00355	00037	00050	00077	00104	00263	00274
WRITA	00153	00086	00176	00281			
WRITB	00177	00113	00200	00270			

```
FTN,L
      PROGRAM UNPKZ,3,80
C
C---- PROGRAM TO UNPACK SEA DATA CASSETTE RECORDS WHICH ARE
C     ON DISC.  ROUTINE FIND SEARCHS FOR A WORD WITH BIT 15
C     (MSB) AND BIT 0 (LSB) SET INDICATING AN END-OR-RECORD
C     MESSAGE WORD.  IF THE NUMBER OF WORDS IN THE RECORD IS
C     CORRECT, THE RECORD IS UNPACKED, OTHERWISE A FLAG IS SET
C     AND THE DATA ELIMINATED.  THE PROGRAM RETURNS THE NUMBER
C     OF OUTPUT DISC RECORDS AND THE VERSION NUMBER.
C
C     PARAMETERS PASSED TO UNPKZ VIA RMPAR
C
C        1. NDSRC - NUMBER OF INPUT DISC RECORDS
C        2. NWDSR - NUMBER OF WORDS PER DISC RECORD, IHED(26)
C        3. NWCAS - NUMBER OF WORDS PER CASSETTE RECORD, IHED(24)
C        4. NWSUB - NUMBER OF WORDS PER UNPACKED SUBRECORD, IHED(52)
C
C     PARAMETERS RETURNED TO TRANZ VIA PRTN
C
C        1. JDSRC - NUMBER OF OUTPUT DISC RECORDS
C        2. IVER  - VERSION NUMBER OF UNPKZ
C
C     PROGRAM UNPKZ USES THE FOLLOWING HP ASSEMBLY ROUTINES
C     WHICH MUST BE PROVIDED AT LOAD TIME:
C
C        FIND, UNPAK, MOVE
C
C     WRITTEN BY D. V. FITTERMAN, U.S.G.S., MAY 1979
C     MODIFIED 28 JANUARY 1980
C
      DIMENSION IPARM(5),JPARM(5),IRDR(4),JBUF(1024),IBUF(1152),
     *IER(5),MASK(4)
      EQUIVALENCE (NDSRC,IPARM(1)),(NWDSR,IPARM(2)),
     *(NWCAS,IPARM(3)),(NWSUB,IPARM(4)),(JDSRC,JPARM(1)),
     *(IVER,JPARM(2))
      DATA LUTTY/1/,LUPRT/6/,LUDSK/10/,
     *IDSRC/0/,JDSRC/1/,ITRK/0/,ISEC/-16/,JTRK/0/,JSEC/0/,JPT/0/,
     *LEFT/0/,JSBRC/0/,IFLGE/0/,IER/5*0/,NPDEL/0/,NCDEL/0/,
     *MASK/2000B,1000B,400B,200B/,LMPRT/100/,NERR/0/,IFGLP/0/,
     *IVER/2/
C
C---- GET PARAMETERS
      CALL RMPAR(IPARM)
C
C---- WRITE OUTPUT HEADER
      CALL EXEC(3,1100B+LUPRT,10)
      WRITE(LUPRT,1000)
 1000 FORMAT(" UNPKZ RECORD LENGTH ERRORS")
C
C---- ADVANCE INPUT POINTERS
   10 CALL NXTRC(ITRK,ISEC,IDSRC)
C
C---- READ A DISC RECORD
      CALL EXEC(1,100B+LUDSK,IBUF(LEFT+1),1024,ITRK,ISEC)
```

```
C
C---- INCREMENT NUMBER OF WORDS LEFT IN INPUT BUFFER
      LEFT=LEFT+NWDSR
C
C---- ZERO INPUT POINTER
      IPT=0
C
C---- SEARCH FOR EOR MARK
   20 CALL FIND(IBUF,IPT,LEFT,IEXIT)
C
C---- DETERMINE IF PRINTING LIMIT IS EXCEEDED
      IF(NERR .LT. LMPRT .OR. IFLGP .EQ. 1) GO TO 40
      WRITE(LUTTY,1010) LMPRT
 1010 FORMAT(" ERRORS EXCEED PRINTING LIMIT OF ",I5/
     *" CONTINUE REPORTING ERRORS? (YE OR NO) _")
      READ(LUTTY,1020) I
 1020 FORMAT(A2)
      IF(I .EQ. 2HYE) GO TO 30
C
C---- SET NO REPORT FLAG
      IFLGP=1
      WRITE(LUPRT,1030) IDSRC,IPT,LEFT
 1030 FORMAT("  IDSRC=",I4," IPT=",I4," LEFT=",I4,
     *"  ERROR REPORTING TERMINATED")
      GO TO 40
C
C---- RESET ERROR COUNTER
   30 NERR=0
C
C---- CHECK FOR NO EOR FOUND
   40 IF(IEXIT .LE. -1) GO TO 60
C
C---- TABULATE READER ERRORS
      DO 50 I=1,4
      IRDR(I)=0
      IF(IAND(IBUF(IPT+IEXIT),MASK(I)) .EQ. MASK(I)) IRDR(I)=1
      IER(I)=IER(I)+IRDR(I)
   50 CONTINUE
C
C---- CHECK NUMBER OF WORDS IN CASSETTE RECORD
      IF(IEXIT .EQ. NWCAS) GO TO 70
C
C---- INCREMENT WRONG LENGTH RECORD COUNTER
   60 NWLRC=NWLRC+1
C
C---- INCREMENT ERROR FLAG
      IFLGE=IFLGE+1
C
C---- INCREMENT PRINTING ERROR COUNTER
      NERR=NERR+1
C
C---- NEVER FOUND AN EOR MARK
      IF(IEXIT .GT. -1) GO TO 80
C---- REPORT ERROR
      IF(IFLGP .EQ. 0) WRITE(LUPRT,1040) IDSRC,IPT,LEFT,NWCAS
```

```
 1040 FORMAT("  IDSRC=",I4," IPT=",I4," LEFT =",I4," NWCAS=",I2,
     *11X,"NO EOR FOUND - DELETED")
C
C---- THROW AWAY ALL DATA
      LEFT=0
      NCDEL=NCDEL+1
      GO TO 100
C
C---- THROW AWAY RECORD DATA
   80 LEFT=LEFT-IEXIT
C
C---- DETERMINE TYPE OF RECORD ERROR
      IF(IEXIT .GT. NWCAS) GO TO 90
C
C---- SHORT RECORD
      IF(IFLGP .EQ. 0) WRITE(LUPRT,1050) IDSRC,IPT,IEXIT,NWCAS,IRDR
 1050 FORMAT("  IDSRC=",I4," IPT=",I4," IEXIT=",I4," NWCAS=",I2,
     *" PSLE=",4I1," SHORT RECORD - DELETED")
C
C---- MOVE INPUT POINTER, DROP ALL DATA
      IPT=IPT+IEXIT
      NCDEL=NCDEL+1
      GO TO 110
C
C---- LONG RECORD
   90 IF(IFLGP .EQ. 0) WRITE(LUPRT,1060) IDSRC,IPT,IEXIT,NWCAS,IRDR
 1060 FORMAT("  IDSRC=",I4," IPT=",I4," IEXIT=",I4," NWCAS=",I2,
     *" PSLE=",4I1," LONG RECORD  - PARTIAL DELETE")
C
C---- MOVE INPUT POINTER, DROP SOME DATA
      IPT=IPT+IEXIT-NWCAS
      LEFT=LEFT+NWCAS
      NPDEL=NPDEL+1
C
C---- UNPACK DATA
   70 CALL UNPAK(IBUF,IPT,JBUF,JPT,IEXIT,MESS)
C
C---- PUT MESSAGE WORD AND IFLGE INTO DATA FLAG WORD
C
C       BIT 0       = DATA FLAG
C       BITS 4-1    = PARITY WORD
C       BITS 8-5    = READER ERRORS
C       BITS 15-9   = # OF RECORDS DROPPED
C
C---- MASK READ MESSAGE WORD AND SHIFT RIGHT 2 BITS
C     SHIFT IFLGE LEFT 9 BITS AND OR WITH MESSAGE WORD
C     OR DATA FLAG AND MODIFIED MESSAGE WORD
      JBUF(JPT+7)=IOR(JBUF(JPT+7),IOR(1000B*IFLGE,IAND(MESS,3770B)/
     *4B))
C
C---- CLEAR ERROR FLAG
      IFLGE=0
C
C---- ADVANCE POINTERS
      IPT=IPT+NWCAS
```

```
      LEFT=LEFT-NWCAS
      JPT=JPT+NWSUB
      JSBRC=JSBRC+1
C
C---- OUTPUT BUFFER FULL?
      IF(JSBRC .LT. 32) GO TO 110
C
C---- WRITE BUFFER TO DISC
      CALL EXEC(2,100B+LUDSK,JBUF,1024,JTRK,JSEC)
C
C---- ADVANCE OUTPUT POINTERS AFTER WRITE
      CALL NXTRC(JTRK,JSEC,JDSRC)
C
C---- ZERO OUTPUT POINTERS
      JSBRC=0
      JPT=0
C
C---- INPUT BUFFER LOW?
  110 IF(LEFT .GE. NWCAS) GO TO 20
C
C---- SHUFFLE REMAINING DATA TO HEAD OF BUFFER
      CALL MOVE(IBUF,IPT,IBUF,0,LEFT)
C
C---- ANY MORE DATA?
  100 IF(IDSRC .LT. NDSRC) GO TO 10
C
C---- CHECK FOR EMPTY OUTPUT BUFFER
      IF(JSBRC .GT. 0) GO TO 120
C
C---- REDUCE OUTPUT BUFFER WORD
      JDSRC=JDSRC-1
      GO TO 150
C
C---- ZERO END OF BUFFER
  120 IF(JPT .GE. 1024) GO TO 140
      DO 130 I=JPT+1,1024
  130 JBUF(I)=0
C
C---- WRITE BUFFER TO DISC
  140 CALL EXEC(2,100B+LUDSK,JBUF,1024,JTRK,JSEC)
C
C---- REPORT RESULTS SUMMARY
  150 IER(5)=IER(1)+IER(2)+IER(3)+IER(4)
      WRITE(LUPRT,1070) NDSRC,JDSRC,NWLRC,NCDEL,NPDEL,IER
 1070 FORMAT(/" UNPKZ COMPLETED: NDSRC=",I5," JDSRC=",I5," NWLRC=",I5/
     *17X," NCDEL=",I5," NPDEL=",I5/
     *17X," PE=",I5," SH=",I5," LO=",I5," EX=",I5," TOTAL=",I5)
      WRITE(LUTTY,1070) NDSRC,JDSRC,NWLRC,NCDEL,NPDEL,IER
C
C---- RETURN PARAMETERS
      CALL PRTN(JPARM)
      STOP
      END
```

FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 03030 COMMON = 00000

```
      SUBROUTINE NXTRC(ITRK,ISEC,IDSRC)
C
C---- SUBROUTINE TO INCREMENT TRACK AND SECTOR ADDRESSES FOR
C     DISC READS AND WRIES.  ALSO INCREMENTS DISC RECORD
C     POINTER.
C
      IF(ISEC .EQ. 80) ITRK=ITRK+1
      ISEC=MOD(ISEC+16,96)
      IDSRC=IDSRC+1
      RETURN
      END
```

```
FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

** NO WARNINGS ** NO ERRORS **   PROGRAM = 00033      COMMON = 00000
```

```
0230          END$
```

```
 PAGE 0001
0001                    ASMB,L,T,C
FIND   R 000004
.ENTR  X 000001
IBUF   R 000000
INPT   R 000001
LENG   R 000002
IEXIT  R 000003
LOOP   R 000013
EX2    R 000026
EX1    R 000027
**  NO ERRORS PASS#1 **RTE ASMB 760924**
```

```
PAGE 0002 #01
0001                      ASMB,L,T,C
0002  00000                      NAM FIND,3,80
0003*
0004* FIND  - ROUTINE TO FIND THE NEXT END OF RECORD
0005*  )
0006                             ENT FIND
0007                             EXT .ENTR
0008  00000 000000  IBUF  BSS 1         ADDRESS OF BUFFER TO BE SEARCHED
0009  00001 000000  INPT  BSS 1         ADDRESS OF IBUF POINTER
0010  00002 000000  LENG  BSS 1         ADDRESS OF LENGTH OF BUFFER TO BE SEA
0011  00003 000000  IEXIT BSS 1         ADDRESS OF EXIT CONDITION
0012*                     = -1  RAN OUT OF DATA AND FOUND NO EOR
0013*                     = >0  # OF WORDS TO EOR
0014*
0015*      MODIFIED 15 JUNE 1979
0016*
0017  00004 000000  FIND  NOP
0018  00005 016001X       JSB .ENTR     RESOLVE INDIRECT ADDRESSES
0019  00006 000000R       DEF IBUF
0020  00007 105745        LDX INPT,I    "X" = INPT
      00010 100001R
0021  00011 105755        LDY =B0       "Y" = ZERO
      00012 000032R
0022  00013 101742  LOOP  LAX IBUF,I    LOAD WORD
      00014 100000R
0023  00015 105770        ISY           INCREMENT "Y"
0024  00016 002031        SSA,SLA,RSS   BIT 15 & BIT 0 = ONE?
0025  00017 002001        RSS           NO, DATA WORD
0026  00020 026027R       JMP EX1       YES, EOR MESSAGE WORD
0027  00021 105760        ISX           INCREMENT "X"
0028  00022 101754        CYA
0029  00023 152002R       CPA LENG,I    "Y" = LENG?
0030  00024 026026R       JMP EX2       YES, NO MORE DATA
0031  00025 026013R       JMP LOOP      NO, KEEP SEARCHING
0032  00026 003401  EX2   CCA,RSS       SET IEXIT = -1
0033  00027 101754  EX1   CYA           SET IEXIT = "Y"
0034  00030 172003R       STA IEXIT,I
0035  00031 126004R       JMP FIND,I
      00032 000000
0036                      END FIND
**  NO ERRORS *TOTAL **RTE ASMB 760924**
```

CROSS-REFERENCE SYMBOL TABLE

.ENTR	00007	00018			
=B0	.....	00021			
EX1	00033	00026			
EX2	00032	00030			
FIND	00017	00006	00035	00036	N
IBUF	00008	00019	00022		
IEXIT	00011	00034			
INPT	00009	00020			
LENG	00010	00029			
LOOP	00022	00031			

```
 PAGE 0001
0001                    ASMB,L,T,C
UNPAK R 000006
.ENTR X 000001
IBUF  R 000000
INPT  R 000001
OBUF  R 000002
OUTPT R 000003
TEXIT R 000004
MESS  R 000005
LOOP  R 000056
IN    R 000074
OUT   R 000103
T1    R 000111
**  NO ERRORS PASS#1 **RTE ASMB 760924**
```

```
                   ASMB,L,T,C
      00000                 NAM UNPAK,3,80
                            ENT UNPAK
                            EXT .ENTR
*
* UNPAK - ROUTINE TO UNPAK SEA DATA CASSETTE RECORDS
*
      00000 000000 IBUF  BSS 1          ADDRESS OF SOURCE BUFFER
      00001 000000 INPT  BSS 1          ADDRESS OF SOURCE BUFFER POINTER
      00002 000000 OBUF  BSS 1          ADDRESS OF DESTINATION BUFFER
      00003 000000 OUTPT BSS 1          ADDRESS OF DESTINATION BUFFER POINTER
      00004 000000 IEXIT BSS 1          ADDRESS OF NUMBER OF WORDS TO UNPACK
      00005 000000 MESS  BSS 1          ADDRESS OF RECORD MESSAGE WORD
      00006 000000 UNPAK NOP
      00007 016001X      JSB .ENTR      RESOLVE INDIRECT ADDRESSES
      00010 000000R      DEF IBUF
      00011 105745       LDX INPT,I     INITIALIZE INDEX REGISTERS
      00012 100001R
      00013 105755       LDY OUTPT,I
      00014 100003R
      00015 006400       CLB
      00016 016074R      JSB IN         I/1
      00017 100045       LSL 5
      00020 016103R      JSB OUT            O/1 MSB CLOCK
      00021 100047       LSL 7
      00022 016074R      JSB IN         I/2
      00023 100050       LSL 8
      00024 016103R      JSB OUT            O/2 LSB CLOCK
      00025 100044       LSL 4
      00026 005000       BLS            MULTIPLY BY 2
      00027 076111R      STB T1
      00030 005020       BLS,BLS        MULTIPLY BY 4
      00031 046111R      ADB T1
      00032 076111R      STB T1         TEN * MSD
      00033 006400       CLB
      00034 016074R      JSB IN         I/3
      00035 100044       LSL 4
      00036 046111R      ADB T1         10*MSD + LSD
      00037 016103R      JSB OUT            O/3 CASSETTE ID#
      00040 100045       LSL 5
      00041 016103R      JSB OUT            O/4 INSTRUMENT #
      00042 100043       LSL 3
      00043 016103R      JSB OUT            O/5 SCAN INTERVAL
      00044 016074R      JSB IN         I/4
      00045 100043       LSL 3
      00046 016103R      JSB OUT            O/6 CHANNELS/SCAN
      00047 100041       LSL 1
      00050 016103R      JSB OUT            O/7 DATA FLAG
      00051 166004R      LDB IEXIT,I
      00052 007004       CMB,INB
      00053 046112R      ADB =B6
      00054 076111R      STB T1
      00055 006400       CLB
      00056 100050 LOOP  LSL 8          UNPACK DATA WORDS
      00057 016074R      JSB IN
      00060 100044       LSL 4
```

```
00061 016103R        JSB OUT
00062 036111R        ISZ T1
00063 026056R        JMP LOOP
00064 100050         LSL 8        UNPACK TEMPERATURE WORD
00065 016074R        JSB IN
00066 100050         LSL 8
00067 016103R        JSB OUT
00070 101742         LAX IBUF,I   LOAD MESSAGE WORD
00071 100000R
00072 172005R        STA MESS,I
00073 126006R        JMP UNPAK,I
00074 000000  IN     NOP          INPUT WORD FROM IBUF
00075 101742         LAX IBUF,I
00076 100000R
00077 105760         ISX
00100 012113R        AND =B77770  MASK 12 BITS
00101 001200         RAL          SHOVE IT TO THE LEFT
00102 126074R        JMP IN,I
00103 000000  OUT    NOP          OUTPUT WORD TO OBUF
00104 105750         SBY OBUF,I
00105 100002R
00106 105770         ISY
00107 006400         CLB
00110 126103R        JMP OUT,I
00111 000000  T1     BSS 1        10*MOST SIG DIG/COUNTER
00112 000006
00113 077770
                     END UNPAK
**  NO ERRORS *TOTAL **RTE ASMB 760924**
```

UNPAK
CROSS-REFERENCE SYMBOL TABLE

```
.ENTR   00004   00015

=B6     .....   00049

=B77770 ....    00068

IBUF    00008   00016   00062   00066   C

IEXIT   00012   00047

IN      00065   00020   00024   00034   00042   00053   00059
        00070

INPT    00009   00017

LOOP    00052   00057

MESS    00013   00063

OBUF    00010   00072

OUT     00071   00022   00026   00037   00039   00041   00044
        00046   00055   00061   00075

OUTPT   00011   00018

T1      00076   00029   00031   00032   00036   00050   00056

UNPAK   00014   00003   00064   00077
```

```
0001                    ASMB,L,T,C
MOVE  R 000005
.ENTR X 000001
SOURC R 000000
SORPT R 000001
DEST  R 000002
DESPT R 000003
NUMBR R 000004
**  NO ERRORS PASS#1 **RTE ASMB 760924**
```

PAGE 0002 #01

```
                  ASMB,L,T,C
      00000               NAM MOVE,3,80
*
* MOVE   - PROGRAM TO MOVE THE CONTENTS OF ONE  ARRAY TO ANOTHER.
                          ENT MOVE
                          EXT .ENTR
      00000 000000  SOURC BSS 1          ADDRESS OF SOURCE BUFFER
      00001 000000  SORPT BSS 1          ADDRESS OF SOURCE BUFFER POINTER
      00002 000000  DEST  BSS 1          ADDRESS OF DESTINATION BUFFER
      00003 000000  DESPT BSS 1          ADDRESS OF DESTINATION BUFFER POINTER
      00004 000000  NUMBR BSS 1          ADDRESS OF NUMBER OF WORDS TO MOVE
      00005 000000  MOVE  NOP
      00006 016001X       JSB .ENTR      RESOLVE  INDIRECT ADDRESSES
      00007 000000R       DEF SOURC
      00010 162004R       LDA NUMBR,I
      00011 002003        SZA,RSS        MOVE ZERO WORDS?
      00012 126005R       JMP MOVE,I     YES, RETURN
      00013 062000R       LDA SOURC      NO, FORM SOURCE BUFFER ADDRESS
      00014 142001R       ADA SORPT,I
      00015 066002R       LDB DEST       FORM DESTINATION BUFFER ADDRESS
      00016 146003R       ADB DESPT,I
      00017 105777        MVW NUMBR,I    MOVE WORDS
      00020 100004R
      00021 000000
      00022 126005R       JMP MOVE,I
                          END MOVE
**  NO ERRORS *TOTAL **RTE ASMB 760924**
```

MOVE
CROSS-REFERENCE SYMBOL TABLE

```
.ENTR   00006   00013
DESPT   00010   00021
DEST    00009   00020
MOVE    00012   00005   00017   00023   00024
NUMBR   00011   00015   00022
SORPT   00008   00019
SOURC   00007   00014   00018
```

```
FTN,L
      PROGRAM EDITZ,3,80
C
C---- PROGRAM TO EDIT GEOMAGNETIC DATA DURING THE TRANSCRIPTION
C     PROCESS.  DATA IS READ FROM DISC AND SCANNED FOR A VARIETY
C     OF ERRORS INCLUDING:
C
C        1. PARITY ERROR (PR)
C        2. INCONSISTENT HEADER (IC)
C        3. SHORT RECORD (SH)
C        4. LOW SIGNAL LEVEL (LO)
C        5. EXCESS DATA (EX)
C        6. IMPROPER CLOCK INCREMENT (CI)
C
C     THE FOLLOWING ACTION IS TAKEN FOR THE ERRORS INDICATED
C     ABOVE:
C
C     PARITY ERROR (PR)
C
C     IF THE FLAG IFLGP WAS SET IN TRANZ, THE DATA ARE KEPT
C     AND THE PARITY ERROR ANNOTATED.  AFTER RETAINING LMPR
C     RECORDS WITH PARITY ERRORS, THE USER IS ASKED IF THE
C     TRANSCRIPTION SHOULD BE TERMINATED.  IF THE USER HAS THE
C     PROGRAM CONTINUE, LMPR IS DOUBLED.  ONLY THE FIRST LPRNT
C     PARITY ERRORS ARE PRINTED.  IF FLAG IFLGP WAS NOT SET,
C     SUBRECORDS CONTAINING PARITY ERRORS ARE REMOVED.
C     NOT SET, SUBRECORDS CONTAINING PARITY ERRORS ARE REMOVED.
C
C     INCONSISTENT HEADER (IC)
C
C     THE HEADER IS CORRECTED TO THE VALUES INPUT TO PROGRAM
C     TRANZ.  WHEN THE NUMBER OF IC ERRORS EXCEEDS LMIC, THE USER
C     IS ASKED IF THE TRANSCRIPTION SHOULD BE TERMINATED.
C
C     SHORT RECORD (SH)
C
C     THESE ERRORS ARE ANNOTATED.
C     IT IS DOUBTFUL THAT ANY ERROR OF THIS TYPE WILL BE
C     ENCOUNTERED SINCE PROGRAM UNPKZ HAS SCREENED THE DATA
C     FOR PROPER RECORD LENGTH.  BUT TO BE ON THE SAFE SIDE....
C
C     LOW SIGNAL (LO)
C
C     THIS ERROR CONDITION IS NOT RELIABLE.  THEREFORE ONLY THE
C     FIRST LMLO OCCURRENCES ARE ANNOTATED.  THE DATA ARE ALWAYS
C     RETAINED.
C
C     EXCESS DATA (EX)
C
C     THIS ERROR CONDITION IS THE LEAST RELIABLE AND IS TREATED
C     THE SAME AS THE LOW SIGNAL CONDITION--THE FIRST LMEX
C     OCCURRENCES ARE ANNOTATED AND ALL DATA ARE RETAINED.
C     THIS CONDITION CAN RESULT FROM A CASSETTE WHICH IS
C     MAGNETICALLY VERY CLEAN.
C
```

```
C     THE FOLLOWING PARAMETERS ARE PASSED TO EDITZ VIA ROUTINE
C     RMPAR BY PROGRAM TRANZ:
C
C       1. NDSRC - NUMBER OF INPUT DISC RECORDS
C       2. IFLGP - PARITY ERROR RETENTION FLAG
C                  =0, REMOVE RECORDS WITH PARITY ERRORS
C                  =1, KEEP RECORDS WITH PARITY ERRORS
C
C     THE FOLLOWING PARAMETERS ARE PASSED TO TRANZ VIA ROUTINE
C     PRTN AFTER COMPLETION:
C
C       1. JDSRC - NUMBER OF OUTPUT DISC RECORDS
C       2. IFLGC - COMPLETION CODE FLAG
C                  =0 SATISFACTORY EDITTING OF DATA
C                  =1 UNSATISFACTORY EDITTING, ABORT TRANSCRIPTION
C       3. IVER  - VERSION NUMBER OF EDITZ
C
C     PROGRAM EDITZ USES THE HEADER INFORMATION STORED IN ARRAY
C     IHED BY PROGRAM TRANZ.  THIS ARRAY IS KEPT IN SYSTEM
C     COMMON.  ARRAY IHED IS 128 WORDS LONG.  PROGRAM EDITZ IS
C     LOADED WITH SYSTEM COMMON USING THE COMMAND:
C
C                  RU,LOADR,99,6,10,0,0
C
C     PROGRAM EDITZ USES THE FOLLOWING HP ASSEMBLY ROUTINES
C     WHICH MUST BE PROVIDED AT LOAD TIME:
C
C       PARWD, MOVE
C
C     WRITTEN BY D. V. FITTERMAN, U.S.G.S., MAY 1979
C     MODIFIED 24 SEPTEMBER 1980
C
      DIMENSION IBUF(1280),JBUF(1024),ICLKA(80),JCLKA(80),
     *JCLKC(80),CLKAI(40),CLKAJ(40),CLKCJ(40),IMISR(40),JMISR(40),
     *DTA(15),DTC(7),IER(7),IPARM(5),JPARM(5)
      COMMON IHED(128)
      EQUIVALENCE (CLKAI,ICLKA),(CLKAJ,JCLKA),(CLKCJ,JCLKC),
     *(NDSRC,IPARM(1)),(IFLGP,IPARM(2)),(JDSRC,JPARM(1)),
     *(IFLGC,JPARM(2)),(IVER,JPARM(3)),(NRATE,IHED(9)),
     *(NWORD,IHED(52)),(NSCAN,IHED(53))
      DATA LUTTY/1/,LUPRT/6/,LUDSK/10/,
     *ITRK/0/,ISEC/-16/,IDSRC/0/,IPTD/0/,IPTC/0/,
     *JTRK/0/,JSEC/0/,JDSRC/1/,JSRC/1/,JPTD/0/,JPTC/8/,
     *LMIC/5/,LMPR/10/,LMLO/10/,LMEX/5/,
     *IFLGC/1/,IFLGD/0/,IFLGH/0/,LEFT/0/,
     *IBUF/1280*0/,CLKAI/40*0.0/,IMISR/40*0/,
     *JBUF/1024*0/,CLKAJ/40*0.0/,CLKCJ/40*0.0/,JMISR/40*0/,
     *NEXT/4/,IFRST/0/,
     *IER/7*0/,IERP/0/,
     *IVER/3/
C
C---- GET PARAMETERS
      CALL RMPAR(IPARM)
C
C---- WRITE HEADER
```

```
      CALL EXEC(3,1100B+LUPRT,10)
      WRITE(LUPRT,1000)
 1000 FORMAT(" EDITZ ERROR REPORT")
C
C---- COMPUTE CLOCK INCREMENT PER SUBRECORD
      DT=FLOAT(NSCAN*2**NRATE)
C
C---- SET JPTC TO NEXT
C     FOR PROPER OPERATION 2 .LE. NEXT .LE. 8
      JPTC=NEXT
C
C---- INPUT DISC RECORD
   10 CALL INPT(ITRK,ISEC,IDSRC,LUDSK,IPTD+NWORD*LEFT,IPTC+LEFT,
     *IBUF,CLKAI,IMISR,NWORD,LEFT)
C
C---- CHECK FOR DROPPED DATA FLAG SET
   20 IF(IFLGD .EQ. 0) GO TO 30
C
C---- INCREMENT MISSING RECORD COUND
      IMISR(IPTC+1)=IMISR(IPTC+1)+IFLGD
C
C---- LOWER DROPPED DATA FLAG
      IFLGD=0
C
C---- START DATA CHECKING
C     CHECK FOR EOF
   30 IF(IBUF(IPTD+4) .EQ. 0 .AND. IDSRC .GE. NDSRC) GO TO 120
C
C---- CHECK FOR LAST RECORD WITH 32 SUBRECORDS
      IF(IPTC .EQ. 0 .AND. IDSRC .GE. NDSRC) GO TO 120
C
C---- CHECK FOR PARITY ERRORS
      CALL PARIT(IPTD,IPTC,IDSRC,NDSRC,LUTTY,LUPRT,IBUF,CLKAI,
     *IMISR,IFLGD,IFLGP,LMPR,IER(2),IEXIT)
      GO TO (40,100,160),IEXIT
C
C---- INCONSISTENT HEADER CHECKING
   40 CALL INCON(IPTD,IPTC,IDSRC,NDSRC,LUTTY,LUPRT,IBUF,CLKAI,
     *IMISR,IHED,IFLGH,LMIC,IER(1),IEXIT)
      GO TO (50,160),IEXIT
C
C---- CHECK FOR SHORT RECORD
   50 CALL SHORT(IPTD,IPTC,IDSRC,LUPRT,IBUF,CLKAI,IMISR,IFLGD,
     *IER(3),IEXIT)
      GO TO (60,60),IEXIT
C
C---- CHECK FOR LOW SIGNAL
   60 CALL LOSIG(IPTD,IPTC,IDSRC,LUPRT,IBUF,CLKAI,LMLO,IER(4))
C
C---- CHECK FOR EXCESS DATA
      CALL EXCES(IPTD,IPTC,IDSRC,LUPRT,IBUF,CLKAI,LMEX,IER(5))
C
C---- CHECK FOR FIRST CLOCK VALUE
      IF(IFRST .EQ. 0) GO TO 80
C
```

```
C---- CHECK CLOCK VALUE
      CALL CHECK(IPTC,JDSRC,JSRC,LUPRT,DT,CLKAI,CLKAJ(JPTC),
     *CLKCJ(JPTC),IMISR(IPTC+1),IMISR(IPTC+2),LEFT,CLKC1,CLKC2,
     *JMIS1,JMIS2,IEXIT)
      GO TO (90,90,80,70),IEXIT
C
C---- NOT ENOUGH DATA TO DO SECONDARY CHECKING
   70 WRITE(LUTTY,1010) IDSRC,IPTC
 1010 FORMAT(/" NOT ENOUGH DATA TO SECONDARY PHASE CHECK: IDSRC=",
     *I4," IPTC=",I2)
C
C---- USER INPUT REQUESTED
   80 CALL UHELP(IPTC,IDSRC,JPTC,JDSRC,JSRC,CLKAI,CLKAJ,CLKCJ,
     *IMISR,JMISR,LEFT,NEXT,LUTTY,LUPRT,DT,IER(6),IFRST,CLKC1,JMIS1,
     *IEXIT)
C
C---- SET IFRST=1
      IFRST=1
C
C---- TEST TYPE OF EXIT: 1=NORMAL, 2=STOP 6 SAVE, 3=STOP & FLUSH
      GO TO (90,120,160),IEXIT
C
C---- FIRST OUTPUT
   90 CALL OUTPT(IPTD,IPTC,JPTD,JPTC,JSRC,JTRK,JSEC,JDSRC,LUDSK,
     *NWORD,NEXT,LEFT,IBUF,JBUF,JCLKA,JCLKC,JMISR,CLKAI,CLKAJ,
     *CLKCJ,CLKC1,JMIS1)
C
C---- SECOND OUTPUT
      IF(IEXIT .NE. 2) GO TO 100
      CALL OUTPT(IPTD,IPTC,JPTD,JPTC,JSRC,JTRK,JSEC,JDSRC,LUDSK,
     *NWORD,NEXT,LEFT,IBUF,JBUF,JCLKA,JCLKC,JMISR,CLKAI,CLKAJ,
     *CLKCJ,CLKC2,JMIS2)
C
C---- INPUT BUFFER LOW?
  100 IF(LEFT .GE. NEXT) GO TO 20
C
C---- MOVE INPUT DATA TO HEAD OF BUFFERS
      CALL MOVE(IBUF,IPTD,IBUF,0,NWORD*LEFT)
      CALL MOVE(ICLKA,IPTC+IPTC,ICLKA,0,LEFT+LEFT)
      CALL MOVE(IMISR,IPTC,IMISR,0,LEFT)
      IPTC=0
      IPTD=0
C
C---- ALL INPUT DISC RECORDS READ?
      IF(IDSRC .LT. NDSRC) GO TO 10
C
C---- MORE DATA TO PROCESS, BUT NO ADDITIONAL INPUT NEEDED
      GO TO 20
C
C---- SATISFACTORY COMPLETION
C     NO MORE DATA, ZERO END OF BUFFER
  120 IF(JPTD .EQ. 0) GO TO 140
      DO 130 I=JPTD+1,1024
  130 JBUF(I)=0
C
```

```
      SUBROUTINE INPT(ITRK,ISEC,IDSRC,LUDSK,IPTD,IPTC,IBUF,
     *CLKAI,IMISR,NWORD,LEFT)
C
C---- ROUTINE TO INPUT AND UNLOAD A DISC RECORD
C
      DIMENSION IBUF(1),CLKAI(1),IMISR(1)
C
C---- ADVANCE INPUT DISC POINTERS
      CALL NXTRC(ITRK,ISEC,IDSRC)
C
C---- READ A DISC RECORD
      CALL EXEC(1,100B+LUDSK,IBUF(IPTD+1),1024,ITRK,ISEC)
C
C---- UNLOAD ACTUAL CLOCK AND MISSING RECORD COUNT
      CALL UNLOD(IPTD,IPTC,NWORD,IBUF,CLKAI,IMISR)
C
C---- INCREMENT DATA LEFT COUNTER
      LEFT=LEFT+32
      RETURN
      END
```

FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 00054 COMMON = 00000

```
C---- WRITE DATA TO DISC
      CALL EXEC(2,100B+LUDSK,JBUF,1024,JTRK,JSEC)
      GO TO 150
C
C---- DECREASE OUTPUT RECORD COUNT IF BUFFER WAS EMPTY
  140 JDSRC=JDSRC-1
C
C---- CLEAR SATISFACTORY COMPLETION FLAG
  150 IFLGC=0
C
C---- TOTAL ERRORS AND TERMINATE PROCESSING
  160 IER(7)=IER(1)+IER(2)+IER(3)+IER(4)+IER(5)+IER(6)
C
C---- WRITE ERROR SUMMARY
      WRITE(LUPRT,1020) JDSRC,(IER(I),I=1,7)
 1020 FORMAT(/" EDITZ COMPLETE: NDSRC=",I4/
     *" IC=",I6," PR=",I6," SH=",I6," LO=",I6," EX=",I6,
     *" CI=",I6," TOTAL=",I6)
      WRITE(LUTTY,1020) JDSRC,(IER(I),I=1,7)
C
C---- RETURN PARAMETERS
      CALL PRTN(JPARM)
      STOP
      END
```

FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 03352 COMMON = 00128

```
      SUBROUTINE NXTRC(ITRK,ISEC,IDSRC)
C
C---- ROUTINE TO INCREMENT TRACK AND SECTOR ADDRESS FOR DISC
C     READS AND WRITES.  ALSO INCREMENTS DISC RECORD POINTER.
C
      IF(ISEC .EQ. 80) ITRK=ITRK+1
      ISEC=MOD(ISEC+16,96)
      IDSRC=IDSRC+1
      RETURN
      END
```

```
FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

**  NO WARNINGS **  NO ERRORS **   PROGRAM = 00033      COMMON = 00000
```

```
      SUBROUTINE PARIT(IPTD,IPTC,IDSRC,NDSRC,LUTTY,LUPRT,IBUF,CLKAI,
     *IMISR,IFLGD,IFLGP,LMPR,IER,IEXIT)
C
C---- PARITY ERROR CHECKING ROUTINE
C
      DIMENSION IBUF(1),CLKAI(1),IMISR(1)
      DATA LPRNT/100/
C
      IEXIT=1
C---- PARITY ERROR CHECK
      IF(IAND(IBUF(IPTD+7),400B) .NE. 400B) RETURN
C
C---- EXTRACT PARITY WORD
      CALL PARWD(IBUF(IPTD+7),MESS)
C
C---- INCREMENT ERROR COUNT
      IER=IER+1
C
C---- WRITE MESSAGE FOR NO MORE PARITY ERROR MESSAGES
      IF(IER .EQ. LPRNT) WRITE(LUPRT,1050)
 1050 FORMAT(" ** NO MORE PARITY ERROR MESSAGES WILL BE PRINTED **")
C
C---- CHECK FOR PARITY ERROR SAVING
      IF(IFLGP .EQ. 0) GO TO 20
C
C---- SAVE PARITY ERROR
      IF(IER .LT. LPRNT) WRITE(LUPRT,1000) IDSRC,IPTC,CLKAI(IPTC+1),
     *MESS
 1000 FORMAT(" PARITY ERROR - DATA RETAINED: IDSRC=",I4," IPTC=",
     *I2," CLKAI=",F8.0," PWORD=",@4)
C
C---- INCREMENT DIFFERENTIAL ERROR COUNT
      IERP=IERP+1
C
C---- TOO MANY ERRORS?
      IF(IERP .LE. LMPR) RETURN
      WRITE(LUTTY,1010) LMPR,IDSRC,IPTC,NDSRC,IER
 1010 FORMAT(/" PARITY ERRORS EXCEEDED DIFFERENTIAL LIMIT (",I4,")"/
     *"  IDSRC=",I4," IPTC=",I2," NDSRC=",I4," TOTAL PR ERRORS=",
     *I5)
C
C---- CHECK IF PROCESSING IS TO CONTINUE
   10 WRITE(LUTTY,1020)
 1020 FORMAT(" CONTINUE PROCESSING? (YE OR NO) _")
      READ(LUTTY,1030) I
 1030 FORMAT(A2)
      IF(I .NE. 2HYE .AND. I .NE. 2HNO) GO TO 10
      IF(I .EQ. 2HNO) GO TO 30
C
C---- ZERO DIFFERENTIAL ERROR COUNT
      IERP=0
C
C---- DOUBLE PARITY ERROR LIMIT
      LMPR=LMPR+LMPR
      RETURN
```

```
C
C---- DELETE PARITY ERROR
   20 IF(IER .LT. LPRNT) WRITE(LUPRT,1040) IDSRC,IPTC,CLKAI(IPTC+1),
     *MESS
 1040 FORMAT(" PARITY ERROR - DATA DELETED : IDSRC=",I4," IPTC=",
     *I2," CLKAI=",F8.0," PWORD=",@4)
C
C---- SET DROPPED DATA FLAG
      IFLGD=IMISR(IPTC+1)+1
C
C---- ELIMINATE DATA - ADVANCE INPUT POINTERS ON EXIT
      IEXIT=2
      RETURN
C
C---- TERMINATE PROCESSING ON EXIT
   30 IEXIT=3
      RETURN
      END
```

FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 00398 COMMON = 00000

```
      SUBROUTINE INCON(IPTD,IPTC,IDSRC,NDSRC,LUTTY,LUPRT,IBUF,
     *CLKAI,IMISR,IHED,IFLGH,LMIC,IER,IEXIT)
C
      DIMENSION IBUF(1),CLKAI(1),IMISR(1),IHED(1)
C
      IEXIT=1
C
C---- INCONSISTENT HEADER CHECK
      MESS=0
      DO 10 I=1,4
      MESS=10B*MESS
      IF(IBUF(IPTD+I+2) .NE. IHED(I+6)) MESS=MESS+1B
   10 CONTINUE
      IF(MESS .EQ. 0) RETURN
C
C---- INCREMENT ERROR COUNT
      IER=IER+1
C
C---- NO MORE MESSAGES FLAG SET?
      IF(IFLGH .EQ. 1) GO TO 40
C
C---- TOO MANY ERRORS?
      IF(IER .GT. LMIC) GO TO 20
C
C---- REPORT INCONSISTENT HEADER
      WRITE(LUPRT,1010) IDSRC,IPTC,CLKAI(IPTC+1),MESS,
     *(IBUF(IPTD+I),I=3,6),(IHED(I),I=7,10)
 1010 FORMAT(" INCORRECT HEADER - CORRECTED: IDSRC=",I4,
     *" IPTC=",I2," CLKAI=",F8.0," HWORD=",@4/
     *13X,"RECORDED  HEADER: ",4(@6,"B ")/
     *13X,"CORRECTED HEADER: ",4(@6,"B "))
      GO TO 40
C
C---- TOO MANY ERRORS
   20 WRITE(LUTTY,1020) LMIC,(IBUF(IPTD+I),I=3,6),(IHED(I),I=7,10),
     *IDSRC,IPTC,NDSRC,IER
 1020 FORMAT(/" HEADER ERRORS EXCEED DIFFERENTIAL COUNT LIMIT (",
     *I3,")"/
     *" RECORDED  HEADER: ",4(@6,"B ")/
     *" CORRECTED HEADER: ",4(@6,"B ")/
     *" IDSRC=",I4," IPTC=",I2," NDSRC=",I4," TOTAL IC ERRORS=",I3)
   30 WRITE(LUTTY,1030)
 1030 FORMAT(" CONTINUE PROCESSING? (YE OR NO) _")
      READ(LUTTY,1040) I
 1040 FORMAT(A2)
      IF(I .NE. 2HYE .AND. I .NE. 2HNO) GO TO 30
C
C---- CHECK FOR TERMINATION
      IF(I .EQ. 2HNO) GO TO 60
C
C---- SET NO MORE MESSAGES FLAG
      IFLGH=1
C
C---- CORRECT HEADER
   40 DO 50 I=1,4
```

```
   50 IBUF(IPTD+I+2)=IHED(I+6)
      RETURN
C
C---- TERMINATE PROCESSING ON EXIT
   60 IEXIT=2
      RETURN
      END
```

FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

** NO WARNINGS ** NO ERRORS **   PROGRAM = 00445   COMMON = 00000

```
      SUBROUTINE SHORT(IPTD,IPTC,IDSRC,LUPRT,IBUF,CLKAI,IMISR,
     *IFLGD,IER,IEXIT)
C
      DIMENSION IBUF(1),CLKAI(1),IMISR(1)
C
      IEXIT=1
C
C---- SHORT RECORD ERROR CHECK
      IF(IAND(IBUF(IPTD+7),200B) .NE. 200B) RETURN
C
C---- INCREMENT SHORT RECORD COUNT
      IER=IER+1
      WRITE(LUPRT,1000) IDSRC,IPTC,CLKAI(IPTC+1)
 1000 FORMAT(" SHORT RECORD - DATA DELETED : IDSRC=",I4," IPTC=",
     *I2," CLKAI=",F8.0)
C
C---- SET DROPPED DATA FLAG
C     IFLGD=IMISR(IPTC+1)+1
C**** PRESENTLY SET OFF
C
C---- ELIMINATE DATA - ADVANCE INPUT POINTERS
      IEXIT=2
      RETURN
      END
```

FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 00094 COMMON = 00000

```
      SUBROUTINE LOSIG(IPTD,IPTC,IDSRC,LUPRT,IBUF,CLKAI,LMLO,IER)
C
      DIMENSION IBUF(1),CLKAI(1)
C
C---- LOW SIGNAL ERROR CHECK
      IF(IAND(IBUF(IPTD+7),100B) .NE. 100B) RETURN
C
C---- INCREMENT ERROR COUNT
      IER=IER+1
C
C---- REPORTING LIMIT EXCEEDED?
      IF(IER .GT. LMLO) RETURN
C
C---- REPORT ERROR
      WRITE(LUPRT,1000) IDSRC,IPTC,CLKAI(IPTC+1)
 1000 FORMAT(" LOW SIGNAL   - DATA RETAINED: IDSRC=",I4," IPTC=",
     *I2," CLKAI=",F8.0)
      RETURN
      END
```

FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 00092 COMMON = 00000

```
      SUBROUTINE EXCES(IPTD,IPTC,IDSRC,LUPRT,IBUF,CLKAI,LMEX,IER)
C
      DIMENSION IBUF(1),CLKAI(1)
C
C---- EXCESS DATA ERROR CHECK
      IF(IAND(IBUF(IPTD+7),40B) .NE. 40B) RETURN
C
C---- INCREMENT ERROR COUNT
      IER=IER+1
C
C---- REPORTING LIMIT EXCEEDED?
      IF(IER .GT. LMEX) RETURN
C
C---- REPORT ERROR
      WRITE(LUPRT,1000) IDSRC,IPTC,CLKAI(IPTC+1)
 1000 FORMAT(" EXCESS DATA  - DATA RETAINED: IDSRC=",I4," IPTC=",
     *I2," CLKAI=",F8.0)
      RETURN
      END
```

FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 00093 COMMON = 00000

```
      SUBROUTINE CHECK(IPTC,JDSRC,JSRC,LUPRT,DT,CLKAI,CLKAJ,
     *CLKCJ,IMSR1,IMSR2,LEFT,CLKC1,CLKC2,JMIS1,JMIS2,IEXIT)
C
C---- CLOCK DIFFERENCE CHECKING ROUTINE
C
      DIMENSION CLKAI(1)
      IEXIT=1
C
C---- SET UP FIRST DIFFERENCE
      CALL DIFF(CLKAJ,CLKAI(IPTC+1),DT,DIFF1,I1,INFL1)
C
C---- TEST FOR DIFF = DT
      IF(I1 .NE. 1 .OR. INFL1 .NE. 1) GO TO 30
      JMIS1=0
      CLKC1=AMD20(CLKCJ+DT)
      RETURN
C
C---- TEST FOR DIFF = I*DT AND 1 .LT. I .LT. 4
C  10 IF(INFL1 .NE. 1 .OR. I1 .LT. 1 .OR. I1 .GE. 4) GO TO 20
C     JMIS1=I1-1
C     CLKC1=AMD20(CLKCJ+DT*FLOAT(I1))
C     RETURN
C
C---- TEST FOR DIFF = DT*(# MISSING RECORD +1)
C  20 IF(DIFF1 .NE. DT*FLOAT(IMSR1+1)) GO TO 30
C     JMIS1=IMSR1
C     CLKC1=AMD20(CLKCJ+DT*FLOAT(JMIS1+1))
C     RETURN
C
C---- TRY SECOND DIFFERENCE CHECKING
C     IS THERE ENOUGH DATA LEFT?
   30 IF(LEFT .GE. 2) GO TO 40
      IEXIT=4
      RETURN
C
C---- SET UP SECOND DIFERENCE
   40 CALL DIFF(CLKAJ,CLKAI(IPTC+2),DT,DIFF2,I2,INFL2)
C
C---- TEST FOR INTEGER SECOND DIFFERENCE AND SPACE FOR CLKC1
      IF(INFL2 .NE. 1 .OR. I2 .LT. 2) GO TO 80
C
C---- TEST FOR SPACE NOT TOO LARGE
      IF(I2 .GT. 6) GO TO 80
      IEXIT=2
C
C---- SET SECOND CLOCK
      CLKC2=AMD20(CLKCJ+DT*FLOAT(I2))
C
C---- TEST FOR CLKC1 POSITION RESOLVABLE BY IMSR1
      IF(IMSR1 .GT. I2-2) GO TO 60
C
C---- DATA AMBIGUITY, USED IMSR1 TO RESOLVE
      JMIS1=IMSR1
      CLKC1=AMD20(CLKCJ+DT*FLOAT(JMIS1+1))
      JMIS2=I2-JMIS1-2
```

```
      WRITE(LUPRT,1000) JDSRC,JSRC,I1,I2,IMSR1,IMSR2
 1000 FORMAT(" CLOCK AMBIGUITY - IMSR1 USED: JDSRC=",I4," JSRC=",I2,
     *" I1=",I6," I2=",I6," IMSR1=",I4," IMSR2=",I4)
      RETURN
C
C---- CHECK FOR CLKC1 POSITION RESOLVABLE BY IMSR2
   60 IF(IMSR2 .GT. I2-2) GO TO 70
C
C---- DATA AMBIGUITY, USED IMSR2 TO RESOLVE
      JMIS2=IMSR2
      JMIS1=I2-JMIS2-2
      CLKC1=AMD20(CLKCJ+DT*FLOAT(JMIS1+1))
      WRITE(LUPRT,1010) JDSRC,JSRC,I1,I2,IMSR1,IMSR2
 1010 FORMAT(" CLOCK AMBIGUITY - IMSR2 USED: JDSRC=",I4," JSRC=",I2,
     *" I1=",I6," I2=",I6," IMSR1=",I4," IMSR2=",I4)
      RETURN
C
C---- SET JMIS1 TO ZERO TO RESOLVE AMBIGUITY
   70 JMIS1=0
      JMIS2=I2-2
      CLKC1=AMD20(CLKCJ+DT)
      WRITE(LUPRT,1020) JDSRC,JSRC,I1,I2,IMSR1,IMSR2
 1020 FORMAT(" CLOCK AMBIGUITY -SET JMIS1=0: JDSRC=",I4," JSRC=",I2,
     *" I1=",I6," I2=",I6," IMSR1=",I4," IMSR2=",I4)
      RETURN
C
C---- PROGRAM CAN'T CORRECT, ASK FOR USER INPUT
   80 IEXIT=3
      RETURN
      END
```

```
FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

** NO WARNINGS ** NO ERRORS **   PROGRAM = 00430      COMMON = 00000
```

```
      SUBROUTINE DIFF(CLKI,CLKJ,DT,DIF,N,INFLG)
C
C---- ROUTINE TO COMPUTE CLOCK DIFFERENCE, INTEGRAL NUMBER OF
C    *DT'S, AND FLAG TELLING IF IT AN INTEGER
C
      DIF=DIF20(CLKI,CLKJ)
      N=DIF/DT
      INFLG=0
      IF(DIF .EQ. DT*FLOAT(N)) INFLG=1
      RETURN
      END
```

FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 00047 COMMON = 00000

```
      SUBROUTINE UHELP(IPTC,IDSRC,JPTC,JDSRC,JSRC,CLKAI,CLKAJ,CLKCJ,
     *IMISR,JMISR,LEFT,NEXT,LUTTY,LUPRT,DT,IER,IFRST,CLKC1,JMIS1,
     *IEXIT)
C
C---- ROUTINE TO INPUT USER CLOCK CORRECTION
C
      DIMENSION CLKAI(1),CLKAJ(1),CLKCJ(1),IMISR(1),JMISR(1),
     *DTA(15),DTC(7)
C
      IEXIT=1
      WRITE(LUPRT,1000) IDSRC,IPTC,CLKAI(IPTC+1)
 1000 FORMAT(" CLOCK  INCREMENTATION ERROR : IDSRC=",I4," IPTC=",
     *I2," CLKAI=",F8.0)
C
C---- INCREMENT ERROR COUNT
      IER=IER+1
      NREST=NEXT
C
C---- CHECK FOR ENOUGH DATA LEFT
      IF(LEFT .GE. NEXT) GO TO 10
      WRITE(LUPRT,1010) IDSRC,IPTC,LEFT,NEXT
 1010 FORMAT(" INSUFFICIENT FUTURE DATA: IDSRC=",I4," IPTC=",
     *I2," LEFT=",I2," NEXT=",I2)
      WRITE(LUTTY,1010) IDSRC,IPTC,LEFT,NEXT
C
C---- SET NREST VALUE
      NREST=LEFT
C
C---- COMPUTE CLOCK DIFFERENCES
C     PAST VALUES
   10 DO 20 I=1,NEXT-1
      II=JPTC+I-NEXT
      DTA(I)=DIF20(CLKAJ(II),CLKAJ(II+1))
   20 DTC(I)=DIF20(CLKCJ(II),CLKCJ(II+1))
      WRITE(LUTTY,1020) IDSRC,IPTC,JDSRC,JPTC,JSRC
 1020 FORMAT(/" CLOCK ERROR: IDSRC=",I4," IPTC=",I2," JDSRC=",
     *I4," JPTC=",I2," JSRC=",I2)
      KK=MOD(JSRC-NEXT+31,32)+1
      LL=JPTC-NEXT+1
      WRITE(LUTTY,1030) KK,CLKAJ(LL),CLKCJ(LL),JMISR(LL)
 1030 FORMAT(/" JSRC",2X,"CLKA",7X,"DTA",6X,"CLKC",7X,"DTC",4X,
     *1X,"MSRC"/
     *1X,I2,1X,F9.0,6X,"-",4X,F9.0,6X,"-",5X,I4)
      DO 30 I=2,NEXT
      KK=MOD(JSRC-NEXT+I+30,32)+1
      LL=JPTC+I-NEXT
   30 WRITE(LUTTY,1040) KK,CLKAJ(LL),DTA(I-1),CLKCJ(LL),DTC(I-1),
     *JMISR(LL)
 1040 FORMAT(1X,I2,1X,F9.0,1X,F9.0,1X,F9.0,1X,F9.0,2X,I4)
C
C---- PRESENT VALUE
      DTA(NEXT)=DIF20(CLKAJ(JPTC),CLKAI(IPTC+1))
C
C---- FUTURE VALUES
      IF(NREST .LE. 1) GO TO 50
```

```
      DO 40 I=NEXT+1,NEXT+NREST-1
      KK=IPTC+I-NEXT
   40 DTA(I)=DIF20(CLKAI(KK),CLKAI(KK+1))
   50 DO 60 I=NEXT+1,NEXT+NREST
      KK=MOD(JSRC-NEXT+I+30,32)+1
      LL=IPTC+I-NEXT
   60 WRITE(LUTTY,1050) KK,CLKAI(LL),DTA(I-1),IMISR(LL)
 1050 FORMAT(1X,I2,1X,F9.0,1X,F9.0,6X,"?",9X,"-",5X,I4)
C
C---- COMPUTE BEST CLOCK ESTIMATE
      CLKC1=AMD20(CLKCJ(JPTC)+DT*FLOAT(IMISR(IPTC+1)+1))
      WRITE(LUTTY,1060) DT,CLKC1
 1060 FORMAT(/" DT=",F5.0,"   BEST CLOCK ESTIMATE=",F9.0/
     *" USE BEST CLOCK ESTIMATE? (YE, NO, OR ST) _")
      READ(LUTTY,1070) I
 1070 FORMAT(A2)
      IF(I .EQ. 2HYE) GO TO 80
      IF(I .EQ. 2HST) GO TO 100
   70 WRITE(LUTTY,1080)
 1080 FORMAT(" CORRECTED CLOCK VALUE? _")
      READ(LUTTY,*) CLKC1
      IF(IFRST .EQ. 0) GO TO 80
C
C---- CHECK FOR DIFFERENCE BEING A MULTIPLE OF DT
      CALL DIFF(CLKCJ(JPTC),CLKC1,DT,DIFF1,JMIS1,INFLG)
      IF(INFLG .EQ. 1 .AND. JMIS1 .GE. 1) GO TO 90
C
C---- WRITE ERROR MESSAGE
      WRITE(LUTTY,1090) DIFF1
 1090 FORMAT(" DIFFERENCE NOT A MULTIPLE OF DT: DIFF=",F9.0)
      GO TO 70
C
C---- FIX UP INTERMEDIATE MISSING RECORD COUNT
   80 JMIS1=IMISR(IPTC+1)
      RETURN
   90 JMIS1=JMIS1-1
      RETURN
  100 WRITE(LUTTY,1100)
 1100 FORMAT(/" STOPPING PROCESSING, SAVE DATA/ (YE OR NO) _")
      READ(LUTTY,1070) I
      IEXIT=2
      IF(I .NE. 2HNO) RETURN
      IEXIT=3
      RETURN
      END


FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


**  NO WARNINGS **  NO ERRORS **   PROGRAM = 00947      COMMON = 00000
```

```
      SUBROUTINE OUTPT(IPTD,IPTC,JPTD,JPTC,JSRC,JTRK,JSEC,JDSRC,
     *LUDSK,NWORD,NEXT,LEFT,IBUF,JBUF,JCLKA,JCLKC,JMISR,CLKAI,CLKAJ,
     *CLKCJ,CLKC,JMIS)
C
C---- ROUTINE TO MOVE OUTPUT TO PROPER AREA AND WRITE TO DISC
C
      DIMENSION IBUF(1),JBUF(1),CLKAI(1),CLKAJ(1),CLKCJ(1),JMISR(1),
     *JCLKA(1),JCLKC(1)
C
C---- PUT ACTUAL AND CORRECTED VALUES IN OUTPUT BUFFER
      CLKAJ(JPTC+1)=CLKAI(IPTC+1)
      CLKCJ(JPTC+1)=CLKC
      JMISR(JPTC+1)=JMIS
C
C---- MOVE DATA WORDS TO OUTPUT BUFFER
      CALL MOVE(IBUF,IPTD,JBUF,JPTD,NWORD)
C
C---- DECOMPOSE CLOCK FOR OUTPUT BUFFER
      JBUF(JPTD+1)=CLKC/32768.
      JBUF(JPTD+2)=CLKC-32768.*FLOAT(JBUF(JPTD+1))
C
C---- SET DATA FLAG TO PROPER DROPPED DATA VALUE
      JBUF(JPTD+7)=IOR(IAND(JBUF(JPTD+7),777B),1000B*JMIS)
C
C---- SET DATA BREAK FLAG
      IF(JMIS .GT. 0) JBUF(JPTD+4)=-IABS(JBUF(JPTD+4))
C
C---- ADVANCE INPUT POINTERS
      IPTD=IPTD+NWORD
      IPTC=IPTC+1
      LEFT=LEFT-1
C
C---- ADVANCE OUTPUT POINTERS
      JPTD=JPTD+NWORD
      JPTC=JPTC+1
      JSRC=JSRC+1
C
C---- OUTPUT BUFFER FULL?
      IF(JSRC .LE. 32) RETURN
C
C---- ZERO END OF BUFFER
      IF(JPTD .GE. 1024) GO TO 20
      DO 10 I=JPTD+1,1024
   10 JBUF(I)=0
C
C---- WRITE DATA TO DISC
   20 CALL EXEC(2,100B+LUDSK,JBUF,1024,JTRK,JSEC)
C
C---- ADVANCE OUTPUT POINTERS
      CALL NXTRC(JTRK,JSEC,JDSRC)
C
C---- SHUFFLE LAST NEXT ACTUAL AND CORRECTED CLOCKS AND DROPPED
C     DATA COUNT TO FRONT OF BUFFERS
      CALL MOVE(JCLKA,64,JCLKA,0,NEXT+NEXT)
      CALL MOVE(JCLKC,64,JCLKC,0,NEXT+NEXT)
```

```
      CALL MOVE(JMISR,32,JMISR,0,NEXT)
C
C---- RESET OUTPUT POINTERS
      JPTD=0
      JPTC=NEXT
      JSRC=1
      RETURN
      END
```

FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 00249 COMMON = 00000

```
      FUNCTION AMD20(CLK)
C
C---- ROUTINE TO REDUCE CLOCK VALUES TO MOD 2*20
      AMD20=CLK
      IF(CLK .GE. 1048576.) AMD20=AMD20-1048576.
      RETURN
      END
```

FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 00028 COMMON = 00000

```
      FUNCTION DIF20(CLKI,CLKJ)
C
C---- ROUTINE TO ADD 2**20 TO DIFFERENCES LESS THAN -1000000.
      DIF20=CLKJ-CLKI
      IF(DIF20 .LT. -1000000.) DIF20=DIF20+1048576.
      RETURN
      END
```

```
FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

**  NO WARNINGS **  NO ERRORS **   PROGRAM = 00032       COMMON = 00000
```

```
      SUBROUTINE UNLOD(IPTD,IPTC,NWORD,IBUF,CLK,MISR)
C
C---- ROUTINE TO UNLOAD CLOCK AND MISSING RECORD COUNT
      DIMENSION IBUF(1),CLK(1),MISR(1)
      IB=IPTD-NWORD
      IC=IPTC-1
      DO 10 I=1,32
      IB=IB+NWORD
      IC=IC+1
      CLK(IC+1)=32768.*IBUF(IB+1)+IBUF(IB+2)
   10 MISR(IC+1)=IAND(IBUF(IB+7),77000B)/1000B
      RETURN
      END
```

FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 00092 COMMON = 00000

```
0758         END$
```

```
 PAGE 0001
0001                    ASMB,L,T,C
PARWD R 000002
.ENTR X 000001
WORD7 R 000000
MESS  R 000001
**  NO ERRORS PASS#1 **RTE ASMB 760924**
```

```
 PAGE 0002 #01
0001                  ASMB,L,T,C
0002  00000                 NAM PARWD,3,80
0003                        ENT PARWD
0004                        EXT .ENTR
0005*
0006*      ROUTINE TO SET OCTAL DIGIT IN MESSAGE WORD CORRESPONDING
0007*      TO THE TRACK WHERE A PARITY ERROR OCCURRED.
0008*      THE PARITY BIT IS ON INDICATES ODD PARITY WHICH IS
0009*      THE NORMAL CONDITION.
0010*
0011*      WRITTEN BY D. V. FITTERMAN, U.S.G.S., JUNE 1979
0012*      MODIFIED 16 JULY 1979
0013*
0014  00000 000000  WORD7 BSS 1          ADDR OF 7TH WORD IN SUBRECORD
0015  00001 000000  MESS  BSS 1          ADDR OF UNPACKED PARITY MESSAGE WORD
0016  00002 000000  PARWD NOP
0017  00003 016001X       JSB .ENTR      RESOLVE INDIRECT ADDRESSES
0018  00004 000000R       DEF WORD7
0019  00005 002400        CLA            SET ALL TRACKS BAD UNTIL TESTED
0020  00006 166000R       LDB WORD7,I    LOAD WORD SEVEN
0021  00007 005310        RBR,SLB        TRACK 4 OK?
0022  00010 032021R       IOR =B0001     YES, TURN ON 1ST OCTAL DIGIT
0023  00011 005310        RBR,SLB        TRACK 3 OK?
0024  00012 032022R       IOR =B0010     YES, TURN ON 2ND OCTAL DIGIT
0025  00013 005310        RBR,SLB        TRACK 2 OK?
0026  00014 032023R       IOR =B0100     YES, TURN ON 3RD OCTAL DIGIT
0027  00015 005310        RBR,SLB        TRACK 1 OK?
0028  00016 032024R       IOR =B1000     YES, TURN ON 4TH OCTAL DIGIT
0029  00017 172001R       STA MESS,I     STORE RESULT
0030  00020 126002R       JMP PARWD,I
      00021 000001
      00022 000010
      00023 000100
      00024 001000
0031                      END PARWD
**  NO ERRORS *TOTAL **RTE ASMB 760924**
```

CROSS-REFERENCE SYMBOL TABLE

```
.ENTR  00004    00017
=B0001 .....    00022
=B0010 .....    00024
=B0100 .....    00026
=B1000 .....    00028
MESS   00015    00029
PARWD  00016    00003    00030    00031
WORD7  00014    00018    00020
```

```
FTN,L
      PROGRAM DBHIZ,3,80
C
C---- PROGRAM TO REPORT DATA BREAK AND CLOCK IRREGULARITIES
C     DURING TRANSCRIPTION.  ALSO DOES HISTOGRAM ANALYSIS FOR
C     EDITTED DATA ON DISC.
C
C     PARAMETERS INPUT TO THIS PROGRAM VIA ROUTINE RMPAR ARE:
C
C       1. NDSRC - NUMBER OF INPUT DISC RECORDS
C
C     PARAMETERS RETURNED VIA ROUTINE PRTN ARE:
C
C       1. NDSRC - NUMBER OF INPUT DISC RECORDS
C       2. IVER  - VERSION NUMBER OF ROUTINE DBHIZ
C
C     PROGRAM DBHIZ USES HEADER INFORMATION STORED IN ARRAY
C     IHED BY PROGRAM TRANZ.  THIS ARRAY IS KEPT IN SYSTEM
C     COMMON.  ARRAY IHED IS 128 WORDS LONG.  PROGRAM DBHIZ IS
C     LOADED WITH SYSTEM COMMON USING THE COMMAND:
C
C                   RU,LOADR,99,6,10,0,0
C
C     WRITTEN BY D. V. FITTERMAN, U.S.G.S., JUNE 1979
C     MODIFIED 11 OCTOBER 1979
C
      DIMENSION IBUF(1024),HIST(224),MAX(7),MIN(7),G(7),IPARM(5),
     *JPARM(5)
      COMMON IHED(128)
      EQUIVALENCE (NDSRC,IPARM(1)),(NRATE,IHED(9)),
     *(NCHAN,IHED(10)),(NWORD,IHED(52)),(NSCAN,IHED(53)),
     *(IVER,JPARM(2))
      DATA LUTTY/1/,LUPRT/6/,LUDSK/10/,
     *IDSRC/1/,ITRK/0/,ISEC/0/,IPT/0/,ISRC/0/,CLKI/0.0/,
     *MAX/7*0/,MIN/7*0/,HIST/224*0.0/,
     *G/3*0.4882813,2*0.0,2*0.4882813/,
     *IVER/1/
C
C---- INPUT PARAMETERS
      CALL RMPAR(IPARM)
C
C---- SET GAINS
      DO 10 I=1,3
      IF(IHED(I+53) .NE. 0) G(I)=IHED(I+53)/2048.
   10 CONTINUE
      DO 20 I=1,2
      IF(IHED(I+56) .NE. 0 .AND. IHED(I+58) .NE. 0)
     *G(I+3)=4882.813/IHED(I+56)/IHED(I+58)
   20 CONTINUE
C
C---- COMPUTE CLOCK INCREMENT
      DT=FLOAT(NSCAN*2**NRATE)
C
C---- WRITE DATA BREAK SUMMARY HEADER
      CALL EXEC(3,1100B+LUPRT,10)
```

```
      WRITE(LUPRT,1000) DT
 1000 FORMAT(" DATA BREAK SUMMARY: DT=",F5.0)
      CALL EXEC(3,1100B+LUPRT,1)
      WRITE(LUPRT,1010)
 1010 FORMAT(8X,"SUB",4X,"LAST",4X,"PRESENT",4X,"MODULAR",3X,"MISS"/
     *3X,"REC",2X,"REC",3X,"CLOCK",5X,"CLOCK",6X,"DIFF",5X,"SREC")
C
C---- READ FIRST RECORD
      CALL EXEC(1,100B+LUDSK,IBUF,1024,ITRK,ISEC)
C
C---- SET MIN AND MAX VALUES
      DO 30 I=1,NCHAN
      MIN(I)=IBUF(I+7)
   30 MAX(I)=MIN(I)
C
C---- START PROCESSING LOOP
C
C---- CHECK FOR END OF FILE
   40 IF(IBUF(IPT+4) .EQ. 0) GO TO 50
C
C---- INCREMENT SUBRECORD COUNTER
      ISRC=ISRC+1
C
C---- CLOCK DIFFERENCE AND DATA BREAK PROCESSING
      CALL DCLOK(IBUF,IPT,IDSRC,ISRC,DT,CLKI,LUPRT)
C
C---- HISTOGRAM PROCESSING
      CALL HISTG(IBUF,IPT,NSCAN,NCHAN,MAX,MIN,HIST)
C
C---- INCREMENT POINTER AND COUNTER
      IPT=IPT+NWORD
C
C---- RECORD PROCESSED?
      IF(ISRC .LT. 32) GO TO 40
C
C---- ANY MORE DATA ON DISC
      IF(IDSRC .GE. NDSRC) GO TO 50
C
C---- ADVANCE DISC POINTERS
      CALL NXTRC(ITRK,ISEC,IDSRC)
C
C---- READ A DISC RECORD
      CALL EXEC(1,100B+LUDSK,IBUF,1024,ITRK,ISEC)
C
C---- RESET POINTERS
      ISRC=0
      IPT=0
      GO TO 40
C
C---- WRITE HISTOGRAM SUMMARY HEADER
   50 CALL EXEC(3,1100B+LUPRT,10)
      WRITE(LUPRT,1020)
 1020 FORMAT(" HISTOGRAM SUMMARY")
C
C---- LOOP OVER CHANNELS
```

```
      KPT=-32
      DO 60 I=1,NCHAN
      KPT=KPT+32
C
C---- GET CHANNEL SUMS
      TOTAL=0.0
      DO 70 J=1,32
   70 TOTAL=TOTAL+HIST(KPT+J)
      FMIN=G(I)*(MIN(I)-2048)
      FMAX=G(I)*(MAX(I)-2048)
      CALL EXEC(3,1100B+LUPRT,1)
      WRITE(LUPRT,1030) I,TOTAL,MIN(I),MAX(I),FMIN,FMAX
 1030 FORMAT(" CHANNEL=",I1,"  NTOT=",F8.0,50X/
     *" MIN=",I4,"  MAX=",I4,"  FMIN=",F7.1,"  FMAX=",F7.1,30X)
      CALL EXEC(3,1100B+LUPRT,1)
      WRITE(LUPRT,1040)
 1040 FORMAT(" BIN  MIN   MAX",4X,"FMIN",5X,"FMAX",6X,"N",7X,"%",20X)
C
C---- SCALE HISTOGRAM VALUES BY GAINS
      MN=-128
      MX=-1
      DO 80 J=1,32
      MN=MN+128
      MX=MX+128
      FMIN=G(I)*(MN-2048)
      FMAX=G(I)*(MX-2048)
      PER=100.*HIST(KPT+J)/TOTAL
   80 WRITE(LUPRT,1050) J,MN,MX,FMIN,FMAX,HIST(KPT+J),PER
 1050 FORMAT(1X,I2,2X,I4,2X,I4,2X,F7.1,2X,F7.1,2X,F8.0,1X,F5.1,20X)
      CALL EXEC(3,1100B+LUPRT,2)
   60 CONTINUE
      CALL EXEC(3,1100B+LUPRT,50)
C
C---- RETURN PARAMETERS
      JPARM(1)=NDSRC
      CALL PRTN(JPARM)
      STOP
      END
```

FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 02232 COMMON = 00128

```
      SUBROUTINE DCLOK(IBUF,IPT,IDSRC,ISRC,DT,CLKI,LU)
C
C---- ROUTINE TO REPORT CLOCK JUMPS AND DATA BREAKS
      DIMENSION IBUF(1024)
C
C---- TEST FOR DATA BREAK
      IDB=2H
      IF(IBUF(IPT+4) .LT. 0) IDB=2HDB
C
C---- COMPUTE PRESENT CLOCK
      CLKJ=32768.*IBUF(IPT+1)+IBUF(IPT+2)
      DIFF=CLKJ-CLKI
      IF(DIFF .LE. -1000000.) DIFF=DIFF+1048576.
C
C---- CHECK VALUE AND DATA BREAK
      IF(DIFF .EQ. DT .AND. IDB .EQ. 2H  ) GO TO 10
C
C---- CLOCK IRREGULARITY
C
C---- EXTRACT NUMBER OF MISSING RECORDS
      MISS=IAND(IBUF(IPT+7),77000B)/1000B
C
C---- REPORT RESULT
      WRITE(LU,1000) IDSRC,ISRC,CLKI,CLKJ,DIFF,MISS,IDB
 1000 FORMAT(2X,I4,2X,I2,3X,F8.0,2X,F8.0,2X,F9.0,2X,I3,3X,A2)
C
C---- SHUFFLE CLOCK VALUES
   10 CLKI=CLKJ
      RETURN
      END
```

FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 00169 COMMON = 00000

```
      SUBROUTINE HISTG(IBUF,IPT,NSCAN,NCHAN,MAX,MIN,HIST)
C
C---- ROUTINE TO DO HISTOGRAM PROCESSING
      DIMENSION IBUF(1024),MAX(7),MIN(7),HIST(224)
      JPT=IPT-NCHAN+7
C
C---- LOOP OVER SCANS
      DO 10 J=1,NSCAN
      JPT=JPT+NCHAN
      KPT=-32
C
C---- LOOP OVER CHANNELS
      DO 10 K=1,NCHAN
      KPT=KPT+32
      IVAL=MAX0(0,MIN0(4096,IBUF(JPT+K)))
C
C---- TEST FOR NEW MAX AND MIN
      IF(IVAL .GT. MAX(K)) MAX(K)=IVAL
      IF(IVAL .LT. MIN(K)) MIN(K)=IVAL
C
C---- PUT COUNT INTO HISTOGRAM BIN
      IBIN=IVAL/128 + 1
   10 HIST(KPT+IBIN)=HIST(KPT+IBIN)+1.0
      RETURN
      END


FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


**  NO WARNINGS **  NO ERRORS **   PROGRAM = 00130      COMMON = 00000
```

```
      SUBROUTINE NXTRC(ITRK,ISEC,IDSRC)
C
C---- ROUTINE TO INCREMENT TRACK AND SECTOR ADDRESS FOR DISC
C     READS AND WRITES.  ALSO INCREMENTS DISC RECORD POINTER.
C
      IF(ISEC .EQ. 80) ITRK=ITRK+1
      ISEC=MOD(ISEC+16,96)
      IDSRC=IDSRC+1
      RETURN
      END
```

FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 00033 COMMON = 00000

```
0214        END$
```

```
FTN,L
      PROGRAM MGAIN,3,80
C----PROGRAM TO CREATE OR UPDATE MAGNETOMETER GAIN VALUES.
C    FIRST RECORD CONTAINS DAY-MONTH-YEAR OF LAST UPDATE.
C     RECORDS 2 THROUGH 32 CONTAIN HX, HY, AND HZ GAIN CONSTANTS
C     IN # OF GAMMAS PER 2048 COUNTS.
C
C
C     WRITTEN BY D. V. FITTERMAN U.S.G.S., AUGUST, 1976.
C
      DIMENSION IDCB(144),IFILE(3),ISIZE(2),IGAIN(3),JGAIN(3)
      DATA LU/1/,IFILE/2HMA,2HGA,2HIN/,ISIZE/1,3/,ITYPE/2/,JGAIN/3*0/
C
C---- TRY TO OPEN THE FILE MAGAIN
      CALL OPEN(IDCB,IER,IFILE,2,518)
      IF(IER .GE. 0) GO TO 100
C
C---- FILE DOESN'T EXIST, CREATE IT ON THE SYSTEM DISC
      CALL CREAT(IDCB,IER,IFILE,ISIZE,ITYPE,518,-2)
      IF(IER .GT. 0) GO TO 10
      WRITE(LU,1000) IER
 1000 FORMAT(" CREATION ERROR=",I5)
      GO TO 999
   10 CALL OPEN(IDCB,IER,IFILE,2,518)
C
C---- INPUT THE CREATION DATE
      WRITE(LU,1010)
 1010 FORMAT(" DATE? (DAY MONTH YEAR) _")
      READ(LU,*) (IGAIN(I),I=1,3)
      CALL WRITF(IDCB,IER,IGAIN,3)
C
C---- INPUT THE MAGNETOMETER GAINS
      WRITE(LU,1020)
 1020 FORMAT(" INPUT GAINS IN GAMMAS/2048 COUNTS  (0 TO STOP))"/
     *" INST #  HX   HY   HZ")
      DO 20 N=1,31
      WRITE(LU,1030) N
 1030 FORMAT(2X,I2,"     _")
      READ(LU,*) (IGAIN(I),I=1,3)
      IF(IGAIN(1) .EQ. 0 .OR. IGAIN(2) .EQ. 0 .OR.
     *IGAIN(3) .EQ. 0) GO TO 30
      NN=N
   20 CALL WRITF(IDCB,IER,IGAIN,3)
C
C---- ZERO THE REST OF THE FILE
   30 IF(NN .EQ. 31)  GO TO 50
      DO 40 I=NN+1,31
   40 CALL WRITF(IDCB,IER,JGAIN,3)
C
C---- PRINT THE FILE
   50 CALL READF(IDCB,IER,IGAIN,3,LEN,1)
      WRITE(LU,1040) (IGAIN(I),I=1,3)
 1040 FORMAT(//" MAGNETOMETER GAINS (GAMMAS/2048 COUNTS)"/
     *" DAY=",I2," MON=",I2," YEAR=",I4/" INST.  HX   HY   HZ")
      DO 60 I=1,31
```

```
      CALL READF(IDCB,IER,IGAIN,3,LEN,I+1)
   60 WRITE(LU,1050) I,(IGAIN(J),J=1,3)
 1050 FORMAT(2X,I2,3X,I4,1X,I4,1X,I4)
C
C---- CLOSE THE FILE
      CALL CLOSE(IDCB)
  999 STOP
C
C---- UPDATE FILE
  100 IFLAG=0
      CALL READF(IDCB,IER,IGAIN,3,LEN,1)
      WRITE(LU,1060) (IGAIN(I),I=1,3)
 1060 FORMAT(" LAST UPDATE: DAY=",I2," MONTH=",I2," YEAR=",I4)
  110 WRITE(LU,1070)
 1070 FORMAT(" INST #? (0 TO STOP) _")
      READ(LU,*) N
      IF(N .EQ. 0) GO TO 150
      IF(N .GT. 31) GO TO 110
C
C---- READ THE RECORD TO BE UPDATED
      CALL READF(IDCB,IER,IGAIN,3,LEN,N+1)
      WRITE(LU,1080) N,(IGAIN(I),I=1,3)
 1080 FORMAT(" INST #=",I3," HX=",I5," HY=",I5," HZ=",I5/
     *" MODIFY? (TYPE CHANGES OR 0'S TO RETAIN)"/
     *"  HX    HY    HZ")
      READ(LU,*) (JGAIN(I),I=1,3)
C
C---- DETERMINE IF THERE WERE ANY CHANGES
      JFLAG=0
      DO 120 I=1,3
      IF(JGAIN(I) .EQ. 0) GO TO 120
      JFLAG=1
      IGAIN(I)=JGAIN(I)
  120 CONTINUE
      IF(JFLAG .EQ. 1) GO TO 130
      GO TO 110
  130 IFLAG=1
C
C---- WRITE THE CHANGES
      CALL WRITF(IDCB,IER,IGAIN,3,N+1)
      CALL RWNDF(IDCB)
      GO TO 110
C
C---- IF NO CHANGES WERE MADE CLOSE THE FILE
  150 IF(IFLAG .EQ. 0) GO TO 50
C
C---- UPDATE THE DATE
      WRITE(LU,1010)
      READ(LU,*) (IGAIN(I),I=1,3)
      CALL WRITF(IDCB,IER,IGAIN,3,1)
      GO TO 50
      END
```

FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 00861   COMMON = 00000

```
0108          END$
```

```
/TRANZ T=00003 IS ON CR00300 USING 00001 BLKS R=0000

:RP,TRANZ
:RP,CASDS
:RP,UNPKZ
:RP,EDITZ
:RP,DBHIZ
:SV,4
:TE, *** BE SURE TO SET SYSTEM CLOCK BEFORE TRANSCRIBING ***
:SV,0
:TR
```