This paper describes a set of programs written for use on Ohio University's 370 computer for reducing and analyzing flight test data.

by

Joseph P. Fischer
Avionics Engineering Center
Department of Electrical Engineering
Ohio University
Athens, Ohio 45701

December 1979

Supported by

National Aeronautics and Space Administration
Langley Research Center
Langley Field, Virginia
Grant NGR 36-009-017
TABLE OF CONTENTS

I   INTRODUCTION                1
II  THE TAPE READER PROGRAM      1
III SUPERVISOR PROGRAM          6
IV  SUBPROGRAM DBREAD           6
V   SUBPROGRAM TDPOS            10
VI  RANGE/AZIMUTH SUBPROGRAM    10
VII PLOTTING                    11
VIII SUMMARY                    11
IX  ACKNOWLEDGEMENTS            11
X  REFERENCES                   13
XI  BIBLIOGRAPHY                13
APPENDIX I  LISTING FOR THE TAPE READER PROGRAM  14
APPENDIX II LISTING FOR THE SUPERVISOR PROGRAM  23
APPENDIX III LISTING FOR SUBPROGRAM DBREAD     26
APPENDIX IV LISTING FOR SUBPROGRAM TDPOS       28
APPENDIX V LISTING FOR BLOCK DATA CONSTANTS REQUIRED BY SUBPROGRAM TDPOS  35
APPENDIX VI LISTING FOR SUBPROGRAM RNGAZ       37
APPENDIX VII LISTING OF SAMPLE PLOTTING PROGRAM 39
I. INTRODUCTION

This paper describes a set of programs designed to be run on the IBM 370/158 computer at Ohio University. These programs are used to read the recorded time differences from the tape produced by the Loran data collection system, convert them to latitude/longitude and produce various plotting input files. The programs have been written so they may be tailored easily to meet the demands of a particular data reduction job. The tape reader program is written in 370 assembler language and the remaining programs are written in standard IBM FORTRAN-IV language. The tape reader program is dependent upon the recording format used by the data collection system and on the I/O macros used at the computing facility. The other programs are generally device-independent, although the plotting routines will be dependent upon the plotting method used.

Figure 1 shows an overall view of the flow of data from the receiver, through the Loran interface unit, to the microprocessor, and the main computing facility. The basic function of the data reduction programs is to convert the recorded data to a more readily usable form; convert the time difference (TD) numbers to latitude/longitude (lat/long), to format a printed listing of the TDs, lat/long, reference times, and other information derived from the data, and to produce data files which may be used for subsequent plotting.

II. THE TAPE READER PROGRAM

Figure 2 shows a functional flow diagram of the tape reader program. Its main function is to find and separate valid data from the tape input. The present format of the recorded tapes is a continuous string of numbers containing the GRI count, and the two TDs. A typical tape record is shown in Figure 3. Although the record format depicted in Figure 3 is eight words of ten bytes each, this is not always the case. Because of the recording operation of the microcomputer, the word positions may be offset to the left or right of the start-record position by a variable number of bytes. It is also possible that an incomplete word was recorded, i.e., one of the three fields (GRI or the TDs) may have been omitted for some reason (usually an interrupt timing problem). Completely invalid data may be included because of initialization problems. The recorded data is in a packed BCD format, as opposed to ASCII or EBCDIC which is more commonly used in the computer. The tape reader program, then, must correctly identify complete data fields, rejecting those which are incomplete or otherwise invalid. The program then converts the data from the packed decimal form to EBCDIC and stores it on a disk file and/or a tape volume. A running count is kept of errors encountered while reading the input tape.

For the discussion which follows, reference is made to the complete program listing in Appendix I. After reading a record from the tape input, a check is made to determine if any errors have been returned from the operating system. An end-of-file (EOF) indication is handled separately and causes the program to close all files and terminate execution (normal exit). If a permanent I/O error is encountered, the record is skipped and the next record is read. A count is maintained of all records skipped in this manner. There are several reasons for permanent I/O errors such as, improper inter-record gaps, or improper recording of the
Figure 1. Flow of Data Through Loran Data Collection System.
Figure 2. Flow Diagram for Tape Reader Program.
Figure 3. Example of GRI-TD Record as Found on Loran Data Tape.
record. The detection and handling of such errors are functions of the tape unit and tape channel used at the computing facility. Any other errors encountered cause an error message to be generated, all files closed, and execution to be terminated.

After a record has been correctly read, the program scans the record for the first occurrence of the TDA mask. Currently, this mask is a hexadecimal "A" in the upper four bits of the most-significant-digit (MSB) of the TDA field. When the TDA mask is found, the digip pointer backspaces four bytes to point to the first byte of the GRI count. Then a check is made for the TDB mask which is the hexadecimal character "B" in the upper four bits of the MSB for the TDB field. After this is verified, a check is made to ensure that there are ten digits in the word. If any of these checks fails, the word is skipped and the next word is tried. A separate count of these errors is kept.

The tape reader program must take into account the fact that part of a word may be on the previous line or on the succeeding line (tape record). If, while checking the word lengths, it is found that some of the first several bytes are missing, the program assumes that these bytes were on the last record. In this case, it assumes that these bytes are already in a temporary buffer which was built during processing of the previous record. Then the remaining bytes on the present record are used to fill out this temporary buffer which is then sent to the format conversion routine. If, while checking the word length, it is found that some of the bytes at the end of the word are missing, the bytes that are present are stored in a temporary buffer and another record is obtained, where it is assumed that the remaining bytes will be found.

When it has been ascertained that a proper ten-byte word exists, a subprogram is called which changes the packed decimal form to a zoned (EBCDIC) format. A check is made to be sure that ten bytes have actually been passed to the conversion routine; also the ten bytes are checked for valid characters. If these checks fail, the word is purged and the invalid word count is increased by one. Before the format conversion takes place, the two TD masks are stripped off (replaced by a four-bit zero). The format conversion is accomplished by taking each four-bit digit (20 in total) and prefixing it with the four-bit hexadecimal "F". Thus, the ten-byte word is converted to a 20-byte EBCDIC word.

The reformatted word is placed in an output buffer. The possible outputs are: a disk file with a dataset name specified when the tape reader program was called, and/or a tape volume. After the output record(s) are written, another pass is made to process another TD data word.

An additional feature built into the data recording system allows event-marks to be identified uniquely by writing an incrementing number into the most significant byte of the GRI count. For no events entered, this MSB digit is set to zero. The tape reader program tests the MSB of the GRI number; if it is non-zero, the entire TD word is converted to zoned format and stored on a separate disk file.
III. SUPERVISOR PROGRAM

The supervisor program is used during the TD-to-position conversion operation. Figure 4 shows a functional diagram of this program. A complete listing is given in Appendix II. The particular version of the supervisor to be discussed here was designed primarily for the CMS Interactive system. To be used in other systems, such as OS/VS, some of the read and write statements and possibly some format statements need be changed. In order to facilitate use by other agencies, I/O operations are confined to this supervisor program; except for subroutine DBREAD, which does its own file-read operations.

The supervisor requests the user to enter parameters for the data reduction job. These parameters, in order of entry, are: the dead-reckoned latitude, dead-reckoned longitude, the number of input data points to be skipped between calculations, the basic Loran chain rate, the reference longitude. The dead-reckoned latitude and longitude are used to initialize the TD-to-position subprogram. The position may be approximate; within one to two degrees of the actual position. Since the conversion program requires a dead-reckoned position each time it is called, the last position calculated is used for the new dead-reckoned position. Because the input file containing the time differences can be quite large, it would take an excessive amount of time to convert every point. If the data is collected at the maximum rate, then there will be approximately ten points every second of real time. Under normal circumstances, one-tenth second represents a very small change in position; thus, the third entry to the supervisor program allows a number of input data points to skipped in between calculations, saving time. The maximum value for this entry is to do one calculation for every 9999th data point. The entry for basic chain rate is used for calculating the time between each set of data points. This is found by multiplying the GRI count read in from the data by the basic chain rate and dividing by 3600, giving the time in hours. This entry is made in decimal format; e.g. if the chain rate is 89700, enter 0.0897. This is the chain rate in fractional seconds. The final two entries are a reference position passed to the range/azimuth routine. Thus the range and azimuth of each calculated point may be found relative to the reference.

After the parameters are read in, the appropriate subprograms are called to read in a data point, convert it to lat/long, and convert it to range/azimuth. The first point is used as a time reference and the GRI count of each succeeding point is used to calculate the time elapsed since the first point. All of the information obtained by the supervisor program is formatted into a listing which is generally printed. Figure 5 shows a sample listing. A separate dataset file is also employed to write the range/azimuth or lat/long, which is then used as input to the plotting programs.

IV. SUBPROGRAM DBREAD

Subprogram DBREAD is used to obtain a GRI count and the two time differences from the input file. The operation of DBREAD is outlined in Figure 6. The complete listing is shown in Appendix III. The present version of this program reads from an 80-byte input file, each record containing four GRI and TD words. Several checks of the numbers read in are performed to minimize problems in other subprograms. One of these checks is a blunder point
Read in Initialization Parameters

Call DBREAD (Get GRI, TDS)

Call TDPOS (TDS to LAT / LONG)

Call RNGAZ (Range and Azimuth)

Calculate Time Elapsed

Format Data for Printed Listing

Write Plot Files

Read Another GRI/TD Record

Figure 4. Flow Diagram for Supervisor Program.
PARAMETERS FOR LORAN DATA REDUCTION PROGRAM:

DEAD RECKONED LATITUDE: 39° 32.22'
DEAD RECKONED LONGITUDE: 82° 08.75'

DATA TAKEN EVERY 1 POINTS.
CHAIN RATE: 0.096
REFERENCE LATITUDE: 39° 13' 22.41'
REFERENCE LONGITUDE: 82° 5' 38.42'

<table>
<thead>
<tr>
<th>COUNT</th>
<th>TDA</th>
<th>TDB</th>
<th>LATITUDE DD MM SS</th>
<th>LONGITUDE DD MM SS</th>
<th>RANGE 4.4 MILES</th>
<th>AZIMUTH DEGREES</th>
<th>TIME HH MM SS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>42594.0</td>
<td>56764.0</td>
<td>39 19 11.92</td>
<td>82 3 44.04</td>
<td>1.6180</td>
<td>95.4288</td>
<td>0 0 0.1</td>
</tr>
<tr>
<td>2</td>
<td>42594.0</td>
<td>56764.0</td>
<td>39 19 11.92</td>
<td>82 4 1.53</td>
<td>1.5707</td>
<td>95.1917</td>
<td>0 0 0.2</td>
</tr>
<tr>
<td>3</td>
<td>42593.9</td>
<td>56763.9</td>
<td>39 19 12.63</td>
<td>82 4 1.53</td>
<td>1.5207</td>
<td>96.3574</td>
<td>0 0 0.3</td>
</tr>
<tr>
<td>4</td>
<td>42594.0</td>
<td>56764.0</td>
<td>39 19 11.92</td>
<td>82 3 44.04</td>
<td>1.5423</td>
<td>98.4251</td>
<td>0 0 0.4</td>
</tr>
<tr>
<td>5</td>
<td>42595.0</td>
<td>56765.0</td>
<td>39 19 19.39</td>
<td>82 4 4.53</td>
<td>1.4796</td>
<td>92.1917</td>
<td>0 0 0.5</td>
</tr>
<tr>
<td>6</td>
<td>42594.0</td>
<td>56764.0</td>
<td>39 19 11.99</td>
<td>82 3 44.06</td>
<td>1.6180</td>
<td>96.4988</td>
<td>0 0 0.6</td>
</tr>
</tbody>
</table>

Figure 5. Example of Printer Listing Produced by Loran Data Reduction Program.
Figure 6. Flow Diagram for Subprogram DBREAD.
trap to delete a number if it differs significantly from the last number read in. Each TD count is held and compared to the new ones read in. If the difference is greater than five microseconds, the new number is deleted and the next number is tested. If the test fails ten consecutive times, the new number replaces the old number used for the comparisons. The purpose of this is to remove one TD reading which is off from the rest for some reason, but to be able to recover if there is a gap in the data because the data collection system was stopped briefly.

Other data checks include a provision to defeat the TD comparator on the first call to DBREAD. An offset may be added to the time differences if desired for analysis of tracking errors. When an end-of-file condition is detected on the input, the GRI count and the time differences are set to zero.

Care should be exercised if this program is modified since this program depends on values from previous calls to determine if any error conditions exist. For this reason, some variables in the program are initialized in a DATA statement and some are initialized in an arithmetic statement.

V. SUBPROGRAM TDPOS

Subprogram TDPOS is a subroutine which converts two time difference numbers to latitude and longitude. This program is a modification of an interactive program already developed for TD conversion. A listing of this program is given in Appendix IV. This program requires that the two TD numbers and the dead-reckoned position be included in the argument list each time it is called. The dead-reckoned position has been discussed briefly in the last section. The further off the dead-reckoned position is from the actual position, the more iterations are necessary to obtain a solution. If the dead-reckoned position is too far off, the calculations may diverge and result in no solution. The number of iterations is kept, and an error condition is noted if the number of iterations becomes too great. Any error conditions are returned in the IERROR argument. IERROR is positive for no error and negative if an error occurs. The computed position is returned in the argument POS.

The chain constants required by TDPOS are provided in a separate BLOCK DATA file. An example is given in Appendix V. Each such file contains the necessary chain data for the master and two selected secondary stations along with the semi-major and semi-minor radii of the earth.

VI. RANGE/AZIMUTH SUBPROGRAM

Subprogram RNGAZ is used to obtain the geodesic arc length and the bearing angle from a given position to a reference position. This program is based on an arc-length computational method used to calculate predicted Loran-C time differences. A complete listing may be found in Appendix VI. The arc length is computed on a reference ellipsoid. This procedure incorporates similar corrections used for Loran-C; i.e., basic shape of the ellipsoid and propagation constants.
Double precision arithmetic is used in this subprogram to maintain accuracy at short arc-lengths. It was found in testing a similar program written in single-precision, that when the arc length was less than several miles, that some of the intermediate results suffered severely from round-off errors. As a result, underflow and overflow problems occurred when the program was run. With the double-precision method, it has been found that no such problems occur and the results are sufficiently accurate at arc-lengths down to one-half mile.

The results returned by this subprogram are in nautical miles for the arc-length and degrees for the bearing. If desired, other units may be easily obtained. Input to this program is the latitude and longitude in decimal degrees.

VII. PLOTTING

Plotting the results of a test flight provides a quick means of analyzing visually the data obtained rather than relying on the printed listing. The listing may be used for accurate point-by-point evaluation, if desired. This section will briefly discuss how the data reduction programs may be used for plotting the data. The plotting routines used in this case are contained in the standard Calcomp plotting package available to FORTRAN users. An example of a FORTRAN-IV program using the range/angle information as input is shown in Appendix VII. A typical plot is shown in Figure 7. The plotting package is quite flexible and allows the user to write a plotting program to satisfy the requirements of the job at hand.

The data reduction programs can produce one or more compact data files which may be used as input for a plotter program. These files usually will only contain the computed latitude and longitude or the range and bearing. Thus these files may be considerably smaller in size than files used to produce them. This makes them easier to store and allow a number of plots on different scales to be produced using the same input file.

VIII. SUMMARY

The data reduction programs consist of several separate programs used for reading Loran data and producing more usable information. The tape reader program may be used separately to obtain formatted time difference numbers. The remaining programs convert the time differences to latitude/longitude and produce plotter input files. These programs have been written so that they may be modified easily to meet the demands of a particular data reduction job.

IX. ACKNOWLEDGEMENTS

These programs were written as an aid to flight test work using Loran-C navigation methods in general aviation. This work is being supported by NASA Grant NGR 36-009-017. Subprogram DBREAD is a modification of a similar program written by Dr. R.W. Lilley.
REFERENCES


BIBLIOGRAPHY

The following publications may prove useful in explaining the computer languages used in this report and how to properly interface these programs to the operating system.


APPENDIX I Listing for the Tape Reader program.
TITLE 'TAPUNF - UNFORMAT PROGRAM FOR LORAN-C DATA RECORDED ON TAPE.'
PRINT ON, GEN, NODATA
SPACE 2

* THIS PROGRAM IS DESIGNED TO READ IN LORAN DATA RECORDED ON MAGNETIC TAPE. IT REMOVES THE TD MASKS AND CHANGES THE DATA FROM PACKED TO ZONED (EBCDIC). THE ZONED DATA MAY THEN BE STORED ON DISK, USING THE FILEID SPECIFIED, AND/OR MAY BE STORED ON ANOTHER TAPE. LORAN EVENT MARKS ARE REMOVED AND STORED ON DISK WITH THE FILEID '(FN) EVENTS C1.'

** **

SPACE 3
LCLA 6 RECLNG DESIRED TAPE LENGTH
&RECLNG SRTA 250
TAPUNF START X'20000'
USING TAPUNF, 12
LR 12, 15
LA 1, 8(1, 1)
CLI 0(1), X'FF'
BE CMDERROR IF SO, THEN ERROR
CLI 0(1), X'00'
B CLI 8(1), C'( ' OPTIONS DELIMITER?
B CLI 8(1), C'( ' OPTIONS?
BNUM +12 YES, CHECK FOR NODISK
LA 1, 8(1, 1)
B NODISKTS CHECK FOR "NODISK"
CLI 16(1), X'00'
BE PUTINA IF NONE, THEN SUBSTITUTE "A"
CLI 16(1), C'( ' OPTIONS?
BE PUTINA MOVE IN DISK "A"
MVC DISKNAME+8(18), 0(1) COMPLETE ID, MOVE TO FSCB
MVC EVNTCTL+8(8), 0(1) FN FOR EVENTS
MVC DISKNAME+25, C' ' STICK IN ADDITIONAL SPACE
LA 1, 24(1, 1)
B OPTCHECK NOW CHECK FOR MODIFIERS
SPACE

PUTINA MVC DISKNAME+8(16), 0(1) MOVE IN PARTIAL ID
MVC EVNTCTL+8(8), 0(1) FN FOR EVENTS
MVI DISKNAME+24, C'A' MOVE IN "A"
MVI DISKNAME+25, C' ' MOVE IN SPACE
LA 1, 16(1, 1)
SPACE

OPTCHECK CLI 0(1), C'( ' SEE IF THERE IS OPTION DELIM.

-15-
THE OPTIONS ALLOWED BY THIS PROGRAM ARE: "IN" TO
CHANGE THE TAPE IN VADDR FROM THE DEFAULT 181 TO ANOTHER;
"OUT" TO CHANGE THE TAPE OUT VADDR FROM THE DEFAULT
182 TO ANOTHER; "NODISK" TO SUPPRESS WRITING THE CONVERTED
DATA ONTO A DISK BUFFER, THIS OPTION MAY ONLY BE USED IF
"TAPE" IS SPECIFIED, EVENTS MARKS ARE STILL WRITTEN;
"TAPE" TO DUMP THE CONVERTED DATA ONTO TAPE AT ADDRESS
182 OR THE MODIFIED ADDRESS.

THE OPTIONS ALLOWED BY THIS PROGRAM ARE: "IN" TO
CHANGE THE TAPE IN VADDR FROM THE DEFAULT 181 TO ANOTHER;
"OUT" TO CHANGE THE TAPE OUT VADDR FROM THE DEFAULT
182 TO ANOTHER; "NODISK" TO SUPPRESS WRITING THE CONVERTED
DATA ONTO A DISK BUFFER, THIS OPTION MAY ONLY BE USED IF
"TAPE" IS SPECIFIED, EVENTS MARKS ARE STILL WRITTEN;
"TAPE" TO DUMP THE CONVERTED DATA ONTO TAPE AT ADDRESS
182 OR THE MODIFIED ADDRESS.

THE OPTIONS ALLOWED BY THIS PROGRAM ARE: "IN" TO
CHANGE THE TAPE IN VADDR FROM THE DEFAULT 181 TO ANOTHER;
"OUT" TO CHANGE THE TAPE OUT VADDR FROM THE DEFAULT
182 TO ANOTHER; "NODISK" TO SUPPRESS WRITING THE CONVERTED
DATA ONTO A DISK BUFFER, THIS OPTION MAY ONLY BE USED IF
"TAPE" IS SPECIFIED, EVENTS MARKS ARE STILL WRITTEN;
"TAPE" TO DUMP THE CONVERTED DATA ONTO TAPE AT ADDRESS
182 OR THE MODIFIED ADDRESS.

THE OPTIONS ALLOWED BY THIS PROGRAM ARE: "IN" TO
CHANGE THE TAPE IN VADDR FROM THE DEFAULT 181 TO ANOTHER;
"OUT" TO CHANGE THE TAPE OUT VADDR FROM THE DEFAULT
182 TO ANOTHER; "NODISK" TO SUPPRESS WRITING THE CONVERTED
DATA ONTO A DISK BUFFER, THIS OPTION MAY ONLY BE USED IF
"TAPE" IS SPECIFIED, EVENTS MARKS ARE STILL WRITTEN;
"TAPE" TO DUMP THE CONVERTED DATA ONTO TAPE AT ADDRESS
182 OR THE MODIFIED ADDRESS.

THE OPTIONS ALLOWED BY THIS PROGRAM ARE: "IN" TO
CHANGE THE TAPE IN VADDR FROM THE DEFAULT 181 TO ANOTHER;
"OUT" TO CHANGE THE TAPE OUT VADDR FROM THE DEFAULT
182 TO ANOTHER; "NODISK" TO SUPPRESS WRITING THE CONVERTED
DATA ONTO A DISK BUFFER, THIS OPTION MAY ONLY BE USED IF
"TAPE" IS SPECIFIED, EVENTS MARKS ARE STILL WRITTEN;
"TAPE" TO DUMP THE CONVERTED DATA ONTO TAPE AT ADDRESS
182 OR THE MODIFIED ADDRESS.

THE OPTIONS ALLOWED BY THIS PROGRAM ARE: "IN" TO
CHANGE THE TAPE IN VADDR FROM THE DEFAULT 181 TO ANOTHER;
"OUT" TO CHANGE THE TAPE OUT VADDR FROM THE DEFAULT
182 TO ANOTHER; "NODISK" TO SUPPRESS WRITING THE CONVERTED
DATA ONTO A DISK BUFFER, THIS OPTION MAY ONLY BE USED IF
"TAPE" IS SPECIFIED, EVENTS MARKS ARE STILL WRITTEN;
"TAPE" TO DUMP THE CONVERTED DATA ONTO TAPE AT ADDRESS
182 OR THE MODIFIED ADDRESS.

THE OPTIONS ALLOWED BY THIS PROGRAM ARE: "IN" TO
CHANGE THE TAPE IN VADDR FROM THE DEFAULT 181 TO ANOTHER;
"OUT" TO CHANGE THE TAPE OUT VADDR FROM THE DEFAULT
182 TO ANOTHER; "NODISK" TO SUPPRESS WRITING THE CONVERTED
DATA ONTO A DISK BUFFER, THIS OPTION MAY ONLY BE USED IF
"TAPE" IS SPECIFIED, EVENTS MARKS ARE STILL WRITTEN;
"TAPE" TO DUMP THE CONVERTED DATA ONTO TAPE AT ADDRESS
182 OR THE MODIFIED ADDRESS.

THE OPTIONS ALLOWED BY THIS PROGRAM ARE: "IN" TO
CHANGE THE TAPE IN VADDR FROM THE DEFAULT 181 TO ANOTHER;
"OUT" TO CHANGE THE TAPE OUT VADDR FROM THE DEFAULT
182 TO ANOTHER; "NODISK" TO SUPPRESS WRITING THE CONVERTED
DATA ONTO A DISK BUFFER, THIS OPTION MAY ONLY BE USED IF
"TAPE" IS SPECIFIED, EVENTS MARKS ARE STILL WRITTEN;
"TAPE" TO DUMP THE CONVERTED DATA ONTO TAPE AT ADDRESS
182 OR THE MODIFIED ADDRESS.
MVC DMS0115A-16(4), 8(1) CHANGE VADDR FOR TAPE COMMAND
LA 1, 16(, 1) POINT TO NEXT OPTION (IF ANY)
B OPTLOOP CHECK FOR MORE
SPACE 2

MVTPOUTD CLI 8(1), C'0' SEE IF NUMERIC OR CHAR.
BL MVTCHECK IF CHAR., BRANCH DOWN
MVC DMS0202A-4+1(3), 8(1) CHANGE VADDR
MVI DMS0202A-4, C'0' INSERT ZERO
LA 1, 16(, 1) POINT TO NEXT OPTION
B OPTLOOP CHECK FOR MORE
SPACE

MVTCHECK MVC DMS0202A-4(4), 8(1) CHANGE VADDR FOR TAPE OUT
LA 1, 16(, 1) POINT TO NEXT OPTION (IF ANY)
B OPTLOOP CHECK FOR MORE
SPACE 2

NODISKTS OI FLAGS, 2 SET NO FILEID BIT
LA 1, 8(, 1) POINT TO NEXT OPTION
B OPTCHECK
SPACE 2

BEGINEX TM FLAGS, 2
B NO RDYRUN IF SET THEN NODISK BIT MUST BE SET
TM FLAGS, 1 SEE IF IT IS
BO RDYRUN IF SET, OK
SPACE
LINEDIT TEXT='DSSTP0052E NO FILEID SPECIFIED, USE NODISK.',
DISP=ERRMSG, DOT=NO, BEND=NO
LA 15, 4 ERROR LEVEL 4
BR 14 RETURN TO CMS
SPACE 2

**************************************************************
A TAPE RECORD IS READ IN AND ANY ERRORS ARE CHECKED. IF
AN "EOT" IS DETECTED, THE PROGRAM WRITES THE INFORMATIONAL
MESSAGES AND LEAVES, IF A PERMANENT I/O ERROR IS DETECTED,
THE BAD RECORD IS SKIPPED OVER AND THE NEXT RECORD IS TRIED.
AFTER THE RECORD IS READ IN, THE TDA MASK IS SEARCHED FOR,
AND WHEN FOUND, BACKSPACED TO THE FIRST BYTE OF THE GRI.
CHECKS ARE MADE TO BE SURE TEN BYTES ARE THERE, IF NOT, THE
MISSING BYTES ARE OBTAINED FROM THE PREVIOUS RECORD, OR ADDED
TO THE NEXT RECORD. THEN THE TEN BYTES OF DATA ARE CONVERTED
TO EBCDIC AND STORED.
**************************************************************

ORIGINAP PAGE IS OF POOR QUALITY
SPACE 2

SLR 7,7 CLEAR GPR7
LA 6, DATACNV SET TO START OF CONVERSION AREA
LA 8, DSKBF INITIALIZE GPR8
LA 9, TPOUTBF INITIALIZE GPR9
LA 5, TPINBF SET ADDRESS OF START OF TAPE BUFFER

RDTAPE TPINBF, ERRCNV READ A RECORD FROM TAPE

LTR 15,15 CHECK THE RETURN CODE
BZ TPCONT IF ZERO, START BYTE MOVE OPERATION
STH 158E15RR STORE RETURN CODE FOR TESTING

CTJ P,1 5RR+1,2 SEE IF POT DETECTED
BF EOT YES, WRAP UP OPERATION
CLI R15RR+1,3 SEE IF I/O ERROR
BE TRYAGAIN IF NOT, CANT CONTINUE
LR 2,15 SAVE RETURN CODE
INFDIT TETT='DSTUP10E ERROR ON READING TAPE.',

DISP=ERRMSG, DOT=NO, RENT=NO
LR 15,2 GET RETURN CODE BACK...
BR 14 ...AND RETURN TO CMS

SPACE 2

TRYAGAIN TAPECTL FSR SKIP OVER BAD RECORD
AP TPERNBR(2), PONE ADD ONE TO ERROR COUNT
B TREAD TRY TO READ ANOTHER RECORD

SPACE 2

TPCONT LTR 7,7 TEST TO SEE IF FILL ON LEFT
BZ NOFILL IF ZERO, NO FILL REQUIRED

FILL MVC 0(1,6),0(5) MOVE A BYTE TO CONVERT AREA
LA 6,1(6) NEXT LOCATION IN CONVERT
LA 5,1(5) NEXT LOCATION TO BE MOVED
BCTR 7,0 ONE LESS TO DO
LTR 7,7 SEE IF IT IS ZERO
BZ CONVHEB YES, COMPLETE FIELD, SO CONVERT
B FILL OTHERWISE, DO IT AGAIN

SPACE

NOFILL LA 3,10(5) LOAD TEN LOCATIONS DOWN...
CL 3, TBPEND ...AND SEE IF OUT-OF-BOUNDS
BCH NOEND IF SMALLER, THEN OK
LA 7,10 TEN BYTE COUNT

DROPRT MVC 0(1,6),0(5) MOVE IN A BYTE
LA 6,1(5) INCREMENT 5
LA 6,1(6) INCREMENT 6
BCTR 7,0 DECREMENT 7
CL 5, TBPEND SEE IF AT EDGE YET
BNP DROPRT NO, CONTINUE MOVING
LA 5, TPINBF BACK TO BEGINNING
B TREAD READ ANOTHER RECORD

SPACE

NOEND SLR 3,3 CLEAR 3 AGAIN
IC 3,0(5) LOAD IN A BYTE
SRL 3,4 GET RID OF LOWER FOUR BITS
CH 3, AMASK SEE IF TDA MASK IS THERE
BE FA YES, FOUND IT
LA 5,1(5) NO, TRY NEXT BYTE
CL 5, TBPEND AT END OF RECORD?
BP TREAD-4 YES, GET ANOTHER RECORD

-18-
**FILE: T**  
**ASSEMBLE C**  
**OHIO UNIVERSITY AVIONICS ENGINEERING CENTER**

<table>
<thead>
<tr>
<th>B</th>
<th>NOEND+2</th>
<th>GO THROUGH LOOP AGAIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL</td>
<td>5, F4</td>
<td>POINT TO GRID FIELD</td>
</tr>
<tr>
<td>MVC</td>
<td>0(10, 6), 0(5)</td>
<td>MOVE THE FIELD TO CONVERT AREA</td>
</tr>
<tr>
<td>LA</td>
<td>6, DATACNVT+10</td>
<td>DO THIS FOR TEST LATER ON</td>
</tr>
<tr>
<td>LA</td>
<td>5, 10(5)</td>
<td>NEXT FIELD TO BE MOVED</td>
</tr>
<tr>
<td>SPACE 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

********************************************************
* CONVERSION IS DONE ONE BYTE AT A TIME. THE BYTES
* ARE CHECKED TO BE SURE THEY ARE IN THE RANGE OF
* ZERO THROUGH NINE: INCLUSIVE. IF THEY ARE NOT, THE
* ENTIRE DATA RECORD IS REJECTED.

********************************************************

| SPACE 2 |         |                       |

| CONVHEB | CL 6, DATACNVT | AT END OF CONVERT BUFFER |
| BNE      | BADRECVD       | IF NOT, BAD RECORD DETECTED |
| LA 6, DATACNVT | POINT TO BEGINNING |
| IC 3, 4(6) | TEST FOR TDA MASK |
| SRL 3, 4 | STRIP OFF LOWER BITS |
| CH 3, AMASK | IS "A"? |
| BNE BADRECVD | NO, ERROR |
| IC 3, 7(6) | TEST FOR TDB MASK |
| SRL 3, 4 | STRIP OFF LOWER BITS |
| CH 3, BMASK | IS "B"? |
| BNE BADRECVD | NO, ERROR |
| MI 4(6), X'0F' | STRIP OFF TDA MASK |
| MI 7(6), X'0F' | STRIP OFF TDB MASK |
| SPACE |         |                       |
| LA 10, CONVERT | START AT BEGINNING |

| CVZONE | IC 3, 0(6) | GET A BYTE |
| SRL 3, 4 | SHIPT OFF LOWER FOUR Bits |
| CH 3, HO | SEE IF NUMBER LESS THAN "0" |
| BNE BADRECVD | IF YES, REJECT RECORD |
| CH 3, H9 | SEE IF NUMBER GREATER THAN "9" |
| BNE BADRECVD | IF YES, REJECT RECORD |
| O 3, ZONEMASK | ADD THE ZONE MASK |
| STC 3, 0(10) | AND PLACE IN CONVERT BUFFER |
| LA 10, 1(10) | NEXT BYTE IN BUFFER |
| IC 3, 0(6) | GET THE SAME CHARACTER |
| N 3, STRIPO4 | STRIP OFF THE UPPER FOUR BITS |
| CH 3, HO | CHECK IF VALID |
| BNE BADRECVD | REJECT RECORD IF NOT |
| CH 3, H9 | CHECK IF VALID |
| BNE BADRECVD | REJECT IF NOT |
| O 3, ZONEMASK | ADD THE ZONE MASK |
| STC 3, 0(10) | STORE IN BUFFER |
| LA 10, 1(10) | NEXT BYTE |
| LA 6, 1(6) | NEXT BYTE TO BE DONE |
| CL 6, DATACNVT | SEE IF DONE |
| BNE CVZONE | NO, CONTINUE THIS LOOP |
| LA 6, 1(10) | LOAD GPR6 |
| SPACE |         |                       |
| TM DATACNVT, X'FP' | CHECK IF FIRST NUMBER IS ZERO |

-19-
FILE: TAPUNF ASSEMBLE C

ORIO UNIVERSITY AVIONICS ENGINEERING CENTER

```
BZ  SEEKSK IF ZERO, NO EVENT ENTERED TAP02760
FSWRITE FSCB=EVTCTL PUT EVENT ON DISK TAP02770
LTR 15,15 CHECK THE RETURN CODE TAP02780
BZ  NOPILL IF ZERO, DO NEXT RECORD TAP02790
SPACE TAP02800
LR 2,15 SAVE RETURN CODE TAP02810
LINEDIT TEXT='DNSTP002E ERROR ON WRITING TO '""LORAN EVENTS C'"" TAP02820
""','FSCB=EVNTCTL PUT EVENT ON DISK TAP02830
LR 15,2 GET RETURN CODE BACK TAP02840
BR 14 AND RETURN TO CMS TAP02850
```

```
SPACE 2

```

```
SEEDSK TM FLAGS, 1 SEE IF WRITING TO DISK TAP02860
BZ  SEEKP IF NOT, CHECK FOR TAPE TAP02870
MVC 0(20,8),CONVERT MOVE THE NUMBER TO DISK BUFFER TAP02880
LA 8,20(,8) NEXT FIELD AREA TAP02890
CL 9,DSKBFEND CHECK IF BUFFER IS FULL TAP02900
BL 9,DSKBF IF NOT, GO ON TAP02910
LA 8,DSKBF RE-INITIALIZE GPR8 TAP02920
FSWRITE FSCB=DISKNAME WRITE TO DISK TAP02930
LTR 15,15 CHECK RETURN CODE TAP02940
BZ  SEEKP IF ZERO, GO ON TAP02950
SPACE TAP02960
LR 2,15 SAVE RETURN CODE TAP02970
LINEDIT TEXT='DNSTP002E ERROR ON WRITING TO ""LORAN EVENTS C"" TAP02980
""','FSCB=EVNTCTL PUT EVENT ON DISK TAP02990
LR 15,2 GET RETURN CODE BACK TAP03000
```

```
SPACE 2

```

```
SEEKP TM FLAGS, 4 CHECK IF TAPE BEING USED TAP03010
BZ  NOPILL TAP03020
MVC 0(20,9),CONVERT MOVE DIGITS TO TAPE BUFFER TAP03030
LA 9,20(,9) POINT TO NEXT FIELD TAP03040
CL 9,TPOUTEND SEE IF BUFFER FULL TAP03050
BNH NOPILL IF NOT, PROCESS ANOTHER FIELD TAP03060
LA 9,TPOUTBP POINT BACK TO BEGINNING TAP03070
WRTAPP (9),800,182 DUMP BUFFER TO TAPE TAP03080
LTR 15,15 CHECK RETURN CODE FROM DUMP TAP03090
BZ  NOPILL IF ZERO, PROCESS ANOTHER RECORD TAP03100
LR 2,15 SAVE RETURN CODE TAP03110
SPACE TAP03120
```

```
SPACE 2

```

```
LINEDIT TEXT='DNSTP002E ERROR ON WRITING TO TAPE.' TAP03130
```

```
SPACE
```

```
LR 15,2 GET RETURN CODE BACK TAP03140
BR 14 GO TO CMS TAP03150
```

```
SPACE 3

```

-20-
```
BADRECD LA 6, DATACNV T  POINT TO BEGINNING OF CONVERT  TAP03310
AP RECE NRR(2), PONE  ADD ONE TO ERROR COUNT  TAP03320
B NOPILL  TRY NEXT FIELD  TAP03330
SPACE 2  TAP03340

READ LA 6, DATACNV T  POINT TO BEGINNING OR CONVERT  TAP03350
*  TAP03360
*   THE PROGRAM COMES HERE AFTER READING A FILE MARK  TAP03370
*   ON THE TAPE. THE COUNTS FOR TAPE RECORDS SKIPPED  TAP03380
*   AND DATA RECORDS REJECTED ARE UNPACKED AND DISPLAYED  TAP03390
*   ON THE TERMINAL. THEN, CONTROL IS RETURNED TO CHS.  TAP03400
*  TAP03410

SPACE 2  TAP03420
NEW UNPK MSG1+23(3), TPERNRBR(2)  CHANGE TO EBCDIC  TAP03430
OI MSG1+25, X'PO'  REMOVE SIGN  TAP03440
UNPK MSG2+23(3), RECE NRR(2)  CHANGE TO EBCDIC  TAP03450
OI MSG2+25, X'PO'  REMOVE SIGN  TAP03460
WRTERM ' '  CARRIAGE RETURN  TAP03470
WA ITT  TAP03480
WRTERM MSG1, 26  TAP03490
WA ITT  TAP03500
WRTERM MSG2, 26  TAP03510
WA ITT  TAP03520
SLR 15, 15  ZERO RETURN CODE  TAP03530
BR 14  BACK TO CHS  TAP03540
EJECT  TAP03550

DS 0D  TAP03560
OPTIN DC CL8'IN'  TAP03570
OPTOUT DC CL8'OUT'  TAP03580
OPTNDSK DC CL8'NODISK'  TAP03590
OPTNTAPE DC CL8'TAPE'  TAP03600
P4 DC P'4'  TAP03610
ZONENASK DC H'0', X'00F0'  TAP03620
STIP04 DC H'0', X'00FF'  TAP03630
DATACNV DC AL4(DATACNV T+10)  TAP03640
TPBFEND DC AL4(TPB END+EREALNG)  TAP03650
DSDKBFEND DC AL4(DSKBF+80)  TAP03660
TPOUTEN D DC AL4(TPOUTBF+300)  TAP03670
DISKNAME PSCB 'REC F=P, BUFFER=DKBF, BSIZ E=80  TAP03680
EVTCTL PSCB 'LORAN EVENTS C1', RECE F=P, BUFFER=CON VERT, BSIZE=20  TAP03690
E15RR DS H  TAP03700
HO DC H'0'  TAP03710
H9 DC H'9'  TAP03720
AMASK DC X'0000A'  TAP03730
BMASK DC X'0000B'  TAP03740
FLAGS DC X'00'  TAP03750
CON VERT DS 21X  TAP03760
TPBF DS 8EREALNG.X  TAP03770
TPOUTBF DS 800X  TAP03780
DSDKBF DS 80X  TAP03790
DATACNV T DC 10C'  TAP03800
DS 10X  TAP03810
TPERRBR DC P1.2'0'  TAP03820
RECE NRR DC P2'0'  TAP03830
PONE DC P1'  TAP03840

ORIGINAL PAGE IS OF POOR QUALITY
<table>
<thead>
<tr>
<th>MSG1</th>
<th>DC</th>
<th>C' TAPE RECORDS REJECTED:</th>
<th>TAP03660</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSG2</td>
<td>LC</td>
<td>C' DATA RECORDS REJECTED:</td>
<td>TAP03670</td>
</tr>
<tr>
<td>END</td>
<td>TAPUNP</td>
<td></td>
<td>TAP03680</td>
</tr>
</tbody>
</table>
APPENDIX II Listing for the Supervisor program.
DIMENSION TOS(2),POS(2),IPOS(2),FPOS(2),APOS(2),MPOS(2),SPOS(2),ORLNO(2)

REAL*8 R LAT,RLONG,P1
COMMON/RNGPOS/RLAT,RLONG,
DATA PI/3.14159265358979/
TIME=9.0
GRIS=0.0
READ 3,IDEG+MIN+SEC+IDEG1+MIN1+SEC1
DRPOS(1)=IDEG1+MIN+SEC/60.+60.
DRPOS(2)=IDEG1+(MIN1+SEC1/60.)/60.
READ 2,LOOPS
READ 1,CRF
READ 3,IDEG+MIN+SEC+IDEG1+MIN1+SEC1
PLAT=(IDEG+(MIN+SEC/60.+60.)*PI/180.*
RLONG=(IDEG1+(MIN1+SEC1/60.)/60.*)*PI/180.*
PRINT 9,DRPOS+LOOPS+CRF,IDEG+MIN+SEC,IDEG1+MIN1+SEC1
PRINT 9
CALL DAREAD(GRI,TDS)
IF(GRI.EQ.0.)GOTO 10
ICOUNT=I
GOTO 14
DO 13 I=1,LOOPS
CALL DAREAD(GRI,TDS)
IF(GRI.EQ.0.)GOTO 10
13 CONTINUE
ICOUNT=ICOUNT+I
14 GRII=GRI-GRIS
GRIS=GRI
TIME=TIME+GRI+CRP/3600.0
CALL TPOS(TDS,POS,DRPOS,IFROP)
IF(IERROR)111,110,11C
110 DO 21 I=1,2
DRPOS(I)=POS(I)
IPOS(I)=POS(I)
FPOS(I)=POS(I)-IFPOS(I)
APOS(I)=FPOS(I)*60.0
MPOS(I)=APOS(I)
21 SPOS(I)=APOS(I)-MPOS(I)*60.0
IHR=TIME
XMIN=(TIME-IHR)*60.0
MIN=XMIN
SCS=(XMIN-MIN)*60.0
CALL RN+AZ(POS,RHO,AZIM)
PRINT 5,ICOUNT,TDS,(IPOS(I)+MPOS(I)+SPOS(I)*I=1,2)+RHO,AZIM,IHR,MIN,ORLNO(2)
WRITE(11,6)PHOWIM
GOTO 10
111 PRINT 7,TDS
GOTO 10
100 STOP
1 FORMAT(F10.0)
2 FORMAT(I4)
3 FORMAT(I4,1X,12,1X,F5.2)
5 FORMAT(1X,I5,8X,2(F7.1X,8X)*2(I4,1X,12,1X,F5.2,8X)*2(F8,4,8X)*2(I2,LPN0,540,1)
>1X,F4.1)
FILE: LR.CMS FORTRAN C
OHIO UNIVERSITY AVIONICS ENGINEERING CENTER

6 FORMAT(3(F4.4))
7 FORMAT(* TDA = *F7.1*H*STUB = *F7.1*T7)*15(*5*): SOLUTION CAI RN03570
8 FORMAT(WHOMET H1* COUNT* T15* T15* T3C* T15* T47* LATTITUDE* T68* LONGITUDE* T68* LONGLAT* L00580
9 UDF* T47* RANGE* T113* AZIMUTH* T129* TIME* T47* DD MM SS* T68* GND00600
0* DD MM SS* T67* MILES* T107** DEGREES* T119** HH MM SS* /
1 LRD03610

2 FORMAT(1 PARANETERS FOR LORAN DATA REDUCTION PROGRAM** ODEAD LRN03620
3 RECKoned LATITUDE: *F9.4** DEAD RECKoned LONGITUDE: *F8.4** DATLNRN03630
4 A TAKEN EVERY *I4** POINTS** CHAIN RATE: *F6.4** REFERENCE LRN03640
5 LATITUDE: *I15*1X*12*1X*F5.2** REFERENCE LONGITUDE: *I14*1X*12*1X*FLRN03650
6 END
7 LRN03660
8 LRN03670
APPENDIX III Listing for subprogram DBREAD.
SUBROUTINE DBREAD(GRI, TDS)
DIMENSION TDS(2), IBUF(4, 4), TDOLD(2), TDOF(2)
DATA ICALL/0/, TDOLD/0., 0., 0., 0./, TDOF/-2., 17. /
NERR=0
7 IF (IPASS .NE. 4) GOTO 2
READ (10, 40, END=5) ((IBUF(I, J), J=1, 3), I=1, 4)
IPASS=0
2 IPASS=IPASS+1
GRI=IBUF(IPASS, 1)
IF (GRI .EQ. 0.) GRI=1.
DO 11 I=1, 2
11 TDS(I)=IBUF(IPASS, I+1)+TDOF(I)
IF (ICALL) 9, 10, 9
10 ICALL=1
DO 12 I=1, 2
12 TDOLD(I)=TDS(I)
IF (ABS(TDOLD(1)-TDS(1)) .GT. 5. .OR. ABS(TDOLD(2)-TDS(2)) .GT. 5.) GOTO 13
DO 8 T=1, 2
8 TDOLD(T)=TDS(T)
RETURN
5 GRI=0.0
DO 6 I=1, 2
6 TDS(I)=0.0
RETURN
13 NERR=NERR+1
IF (NERR .LT. 10) GOTO 7
NERR=0
ICALL=0
GOTO 7
40 FORMAT (4(I8, I6, I6))
END
-27-
APPENDIX IV Listing for subprogram TDPOS.
SUBROUTINE TDPOS (TH, POS, DRPOS, IERROR)

DIMENSION TOSV (2), JOV (2), ANG (8), AD (8), DN (8), CS (8), POS (2), ZWD (2), PTDP00020
:WM (2), DEL (2), TDR (2), DRD (2), DRM (2), TH (2), A (11), B (11), C (11), D (11), E (11), F (11), G (11), H (11), I (11), J (11), K (11), L (11), M (11), N (11), O (11), P (11), Q (11), R (11), S (11), T (11), U (11), V (11), W (11), X (11), Y (11), Z (11)

:RDM (2), RDR (2), RHP (2), DEL (2), DEL (2), BETA (2), ONG (2), IW (2), ITDP00040
:RD (2), nRPOS (2)

DATA A1/24.0305/, A2/-0.40758/, A3/3.46776E-3/, B1/0.510483/, B2/-0.0139987/, C1/1.1402/, B3/0.001760/, RD/1.745329E-2/, RM/2.908882E-4/, RS/4.848137E-6/

C - COMMON/CHAIN/DEL, A5, A6, AD, DM, CS

C - BEGIN TIME DIFFERENCE TO POSITION CONVERSION.

C -

DO 1 T=1,2
IDRD (I) = DRPOS (I)
DRD (I) = IDRD (I)
1 DRM (I) = (DRPOS (I) - DRD (I)) * 60.0
IERROR = 1
A10 = (A5 * A5 - A6 * 16) / (A5 * A5)
A14 = 1.0 - A6 / A5
A50 = (1.0 + A14 * A14 * A14)
A51 = (A50 - 1.0)
A52 = (A14 * A14) / 2.0
A53 = - A51 / 2.0
A54 = (A14 * A14) / 16.0
A55 = (A14 * A14) / 8.0
A56 = A14 * A14
A57 = A56 * 1.25
A58 = A56 / 4.0
DO 128 K=1,8
IF (AD (K)) 124, 126, 126
124 ANG (K) = RD * AD (K) - RM * DM (K) - RS * CS (K)
GO TO 128
126 ANG (K) = RD * AD (K) + RM * DM (K) + RS * CS (K)
128 CONTINUE
A12 = (ANG (1) - ANG (5) + ANG (2) - ANG (6))
A12 = ABS (A12)
IF (A12 < 0.00001) 7, 7, 8
7 A11 = - 1.0
GO TO 9
8 A11 = 1.0
GO TO 9

C - APPROXIMATE POSITIONS AND STATION COORDINATES.

C -

9 E (1) = ANG (1)
E (2) = ANG (2)
CC (1) = ANG (3)
CC (2) = ANG (4)
E (3) = SIN (E (1))
E (4) = COS (E (1))
E (5) = F (3) / E (4)
E (6) = (F (5)) * (1.0 - A14)
A62 = ATAN (A6)
E (6) = SIN (A62)
E(7) = COS(A62)  
CC(3) = SIN(CC(1))  
CC(4) = COS(CC(1))  
CC(5) = CC(3) / CC(4)  
CC(8) = (CC(5)) * (1.0 - A14)  
A62 = ATAN(CC(8))  
CC(6) = SIN(A62)  
CC(7) = COS(A62)  
I = 1  
GO TO 500  
15 E(9) = A35  
E(10) = A46  
E(11) = A47  
DO 17 J = 1, 11  
A(J) = E(J)  
17 B(J) = CC(J)  
E(1) = ANG(5)  
E(2) = ANG(6)  
CC(1) = ANG(7)  
CC(2) = ANG(8)  
E(3) = SIN(F(1))  
E(4) = COS(F(1))  
E(5) = E(3) / E(4)  
E(8) = (E(5)) * (1.0 - A14)  
A62 = ATAN(E(8))  
E(6) = SIN(A62)  
E(7) = COS(A62)  
CC(3) = SIN(CC(1))  
CC(4) = COS(CC(1))  
CC(5) = CC(3) / CC(4)  
CC(8) = (CC(5)) * (1.0 - A14)  
A62 = ATAN(CC(8))  
CC(6) = SIN(A62)  
CC(7) = COS(A62)  
I = 2  
GO TO 500  
19 E(9) = A35  
E(10) = A46  
E(11) = A47  
DO 21 J = 1, 11  
C(J) = E(J)  
21 D(J) = CC(J)  
TM(1) = A(17) + A(11)  
TM(2) = C(10) + C(11)  
DO 45 M = 1, 2  
BETA(M) = TM(M)  
BEDEL(M) = BETA(M) + DFL(M)  
45 BLEI(M) = BETA(M) + BEDEL(M)  
IQSV(1) = 99999  
IQSV(2) = 99999  
ITER = 0  
82 SDR = DRD(1) + DRD(2) + DRD(2)  
IP(SDR) = 83, 84, 83  
83 DO 30 K = 1, 2  
IP(DRD(K)) = 32, 34, 34
3. \( R\text{ADR} (K) = RD \times DRD (K) - RN \times DRN (K) \)
   GO TO 30

34 \( R\text{ADR} (K) = RD \times DRD (K) + RN \times DRN (K) \)

30 CONTINUE
   E (1) = R\text{ADR} (1)
   E (2) = R\text{ADR} (2)
   A28 = -1.0

84 E (3) = \( \sin (E (1)) \)
   E (4) = \( \cos (E (1)) \)
   E (5) = \( (E (3)) / E (4) \)
   E (8) = \( (E (5)) \times (1.0 - A14) \)
   A62 = \( \tan (E (8)) \)
   E (6) = \( \sin (A62) \)
   E (7) = \( \cos (A62) \)
   DO 86 J = 1, 8

86 CC (J) = N (J)
   I = 3
   GO TO 500

90 C1 = A35
   C2 = A44
   C3 = A45
   C101 = A47
   DO 92 J = 1, 8

92 CC (J) = C (J)
   I = 4
   GO TO 500

95 C4 = A35
   C5 = A44
   C6 = A45
   C104 = A47
   DO 50 J = 1, 8

50 CC (J) = B (J)
   I = 5
   GO TO 500

55 C7 = A35
   C8 = A44
   C9 = A45
   C107 = A47
   IF (A11) 52, 99, 53

53 C10 = C7
   C11 = C8
   C12 = C9
   C110 = C107
   DO 63 J = 1, 8

63 CC (J) = A (J)
   I = 6
   GO TO 500

65 C7 = A35
   C8 = A44
   C9 = A45
   C107 = A47
   C13 = TH (2) - C (12) - C (11) - C101 + C104 - DEL (2)
   C17 = C13 * A4
   C18 = TH (1) - A (13) - A (11) - C110 + C107 - DEL (1)
   C22 = C18 * A4

-31-
C23 = C1 - C17
C24 = C4
C25 = C7 + C22
C26 = C13
C27 = (C2 - C5) * (C25 - C26) + (C23 - C24) * (C11 - C8)
C29 = (C2 - C5) * (C12 - C9) + (C3 - C6) * (C8 - C11)
C30 = C27 / C29
C28 = (C23 - C24 + C30 * (C3 - C6)) / (C5 - C2)
GO TO 130

52 C13 = TH(2) - C(10) - C(11) - C101 + C104 - DEL(2)
C17 = C13 * A4
C18 = TH(1) - A(10) - A(11) - C107 + C104 - DEL(1)
C22 = C18 * A4
C23 = C1 - C17
C24 = C4
C25 = C7 - C22
C27 = (C2 * (C25 - C24) + C21 * (C5 - C8) + C8 * C24 - C5 * C25)
C29 = (C2 * (C6 - C9) + C3 * (C8 - C5) + C5 * C9 - C8 * C6)
C30 = C27 / C29
C28 = (C23 - C24 + C30 * (C3 - C6)) / (C5 - C2)

130 C31 = (A5 * A4 * (1.0 - A10)) / (1.0 - A10 * E(3) * E(3)) ** 1.5
C32 = (A5 * A4 / (1.0 - A10 * E(3) * E(3)) ** 0.5
C33 = (C10 / C31)
C34 = (-C28 / (C32 * E(4)) )
E(1) = E(1) + C33
E(2) = E(2) + C34
IF (A28) 132, 99, 134
132 A28 = 1.0
GO TO 84
C - CONVERSION DONE, RETURN TO DISTANCE-BEARING ROUTINE.
C -

900 IF (IQSV(1).NE. IWD(1)) GO TO 7713
IF (IQSV(2).NE. IWD(2)) GO TO 7713
IF (ABS(OSV(1) - PWN(1)).GT. 0.1) GO TO 7713
IF (ABS(OSV(2) - PWN(2)).GT. 0.1) GO TO 7713
IDR(1) = IDR(1) + 10
DO 839 T = 1, 2
ZTWD(I) = IWD(I)
839 POS(I) = (ZTWD(I) + PWN(I)) / 60.0
RETURN
C - CONTINUE ITERATIONS.
C -

7713 DO 7712 M = 1, 2
DRD(M) = 0.0
DRM(M) = 0.0
OSV(M) = PWN(M)
7712 IQSV(M) = IWD(M)
ITER = ITER + 1
IF (ITER.LT. 100) GO TO 82
ITER = -1
RETURN
134 OMG(1) = E(1)
OMG(2) = E(2)
DO 4840 M=1,2
W=ONG (M)/RD
IWD (N) = W
PFD=IWD (M)
DWD=W-PFD
EWN=DWD*60.0
PWN (M) = ABS (EWN)

4810 PWN (M) = 0.0

4820 IWD (M) = IWD (M) -1
GO TO 4840

4830 IWD (M) = IWD (M) +1
4840 CONTINUE
GO TO 900

C -
C - CALCULATION OF INVERSE VARIABLES.
C -

500 A59=-CC (2)
A60=-E (2)
C35=A59-A60
C36=ABS (C35)

502 A16=2.0*PI-C36
GO TO 505

501 A16=C36

505 IF (A16) 506, 507, 506

507 A16=0.00000005

506 A17=STN (A16)
A18=COS (A16)
A19=R (6) *CC (6)
A20=R (7) *CC (7)
A21=A19+A20*A18
A22= ((A17*CC(7)) **2+(CC (6) *E (7) -E (6) *CC (7) *A18) **2) **0.5
A23=(A20*A17)/A22
A24=1.0-A23*A23
A25=ARSTN (A22)
A26=A25*A25
A27=1.0/A22
A28=A21/A22
A29=A24*A24
A30=(A50*A25)+A19*(A51*A22-A52*A26*A27)
A31=A24*(A53*A25+A53*A22*A21+A52*A26*A28)
A32=A19*A19+(-A52*A21*A22)
A33=A29*(A54*A25+A54*A21+A22-A52*A26*A28-A55*A22*(A21**3))
A34=A19*A24*(A52*A26*A27+A52*A22*A21*A21)
A35=(A30+A31+A32+A33+A34)*A6*A4
A36=(A51*A25)+A19*(-A52*A22-A14*A14*A26*A27)
A37=A24*(-A57*A25+A58*A22*A21+A14*A14*A26*A28)
A38=(A36+A37)*A23+A16
A39=STN (A38)
A40=COS (A38)
A41=(CC (6) *E (7) -A40*E (6) *CC (7) )/(A39*CC (7) )

IF (A41) 510, 509, 510

509 A41=0.00000005
510 A42=1.0/A41
A43=ATAN(A42)
   IF (C35) 515, 514, 514
514 IF (C35-PI) 511, 512, 512
511 IF (A41) 520, 521, 521
520 A43=PI+A43
   GO TO 521
512 IF (A41) 517, 518, 518
515 IF (C35+PI) 511, 511, 516
516 IF (A41) 517, 518, 518
517 A43=PI-A43
   GO TO 521
518 A43=2.0*PI-A43
521 A43=A43+PI
   A43=A43-2.0*PI
   IF (A43) 522, 522, 523
522 A43=A43+2.0*PI
523 A44=SIN(A43)
A45=COS(A43)
A46=A35/1609.344
   IF (A46=100.0) 525, 526, 526
525 A47=B1/A46+B2*A3*A46
   GO TO 527
526 A47=A1/A46+A2+A3-A46
527 A46=A35/A4
   GO TO (15, 19, 93, 95, 55, 65), T
99 RETURN
END
APPENDIX V Lisitng for Block Data constants required by subprogram TDPOS.
BLOCK DATA

C - THIS IS THE CHAIN DATA FOR THE 9960 (US NORTHEAST) STATIONS W (CARIBOU) AND X (NANTUCKET).
C - MODIFIED CONSTANTS TO NORTH AMERICAN DATUM 1927 (NAD-27).

COMMON/CHAIN\$DEL (2), A5, A6, AD (8), DM (8), CS (8)
DATA DEL/11.0E3, 25.0E3/
DATA A5/2.1275406E4, A6/2.1203281E4/
DATA AD/42.0, 76.0, 46.0, 67.0,
> 42.0, 76.0, 41.0, 69.0/
DATA DM/42.0, 49.0, 48.0, 55.0,
> 42.0, 49.0, 15.0, 58.0/
DATA CS/50.47, 34.44, 27.86, 39.16,
> 50.47, 34.44, 11.99, 40.51/
END
APPENDIX VI Listing for subprogram RUGAE.
SUBROUTINE RNGA(ZXPOS, RHO, AZIM)

IMPLICIT REAL*8(A-H, I-Z)
REAL*4 POS(2), RHO, AZIM

COMMON/RNGPOS/PHI, XLNG1

DATA A/6.378206416/ , F/3.1900751040-3/ , PI/3.141592653589793/

PHI=POS(1) * PI/180.
XLNG2=POS (2) * PI/180.
DXLNG=XLNG1-XLNG2
BETA=DATAN ((1. - F) * DTAN (PHI))

BETAI=DATAN ((1. - F) * DTAN (PHI))

C1=DCOS (BETAI) * DSIN (DXLNG)
C2=DCOS (BETA) * DSIN (BETAI) - DSIN (BETA) * DCOS (BETAI) * DSIN (DXLNG)
C3=DSIN (BETA) * DSIN (BETAI) + DCOS (BETA) * DCOS (BETAI) * DSIN (DXLNG)

PSI=DATAN (C1/C2)

THETA=DATAN ((C2*DCOS (PSI) + C1*DSIN (PSI)) / C3)

XN=DSIN (BETA) * DSIN (BETAI) **2

XU=(DSIN (BETA) - DSIN (BETAI)) / DSIN (THETA) **2

XV=(1. - DCOS (THETA) ) / DSIN (THETA) * (THETA-DSIN (THETA)) / DSIN (THETA)

RHO=DABS (A*THETA-A*F*(XN*XU+XN*XV)/4.) / 1852.0

IF (C1.GE.0. .AND. C2.LE.0.) PSI=PSI+180. / PI

IF (C1.LE.0. .AND. C2.GE.0.) PSI=(PI+PSI) * 180. / PI

IF (C1.LE.0. .AND. C2.LE.0.) PSI=(PI+PSI) * 180. / PI

AZIM=PSI
RETURN
END
APPENDIX VII Listing of sample plotting program.
DATA PI/3.14159265/,,XORG/9.5/,,YORG/0.25/,,XSCL/5./,,YSCL/3.125/,,YESBH000010
>3HTPS/,,I3G/,-1/
PRINT 60
READ 61,,IANS
IF (IANS.EQ. YES) TGO=1
CALL PLOTS (BUF,,11)
CALL SYMBOL (XORG,,YORG,,0.25,,3,,0.,-1)
DO 20 T=1,,9
20 CALL SYMBOL (XORG-T,,YORG,,0.25,,3,,0.,-1)
DO 21 T=1,,5
21 CALL SYMBOL (XORG,,YORG+1,,6*T,,0.25,,3,,0.,-1)
CAL SYMBOL (XORG-0.9994/SXSCL,,YORG+0.8092/YSCL,,0.35,,1,,0.,-1)
CALL SYMBOL (XORG+1.184/SXSCL,,YORG+1.3814/YSCL,,0.35,,1,,0.,-1)
CALL SYMBOL (XORG-4.5399/SXSCL,,YORG+24.7055/YSCL,,0.35,,1,,0.,-1)
CALL SYMBOL (XORG-14.3832/SXSCL,,YORG+21.7253/YSCL,,0.35,,1,,0.,-1)
CALL SYMBOL (XORG-32.7287/SXSCL,,YORG+21.1557/YSCL,,0.35,,1,,0.,-1)
CALL SYMBOL (XORG-45.2794/SXSCL,,YORG+20.5569/YSCL,,0.35,,1,,0.,-1)
CALL SYMBOL (XORG-21.6659/SXSCL,,YORG+6.0122/YSCL,,0.35,,1,,0.,-1)
C
11 READ (10,,1,,END=100) RNG,,AZM
RNGS=RNG
AZMS=AZM
RHOX=RNG*SIN (AZM*PI/180.)/XSCL
RHOY=RNG*COS (AZM*PI/180.)/YSCL
CALL PLOT (XORG+RHOX,,YORG+RHOY,,2)
GOTO 10
C
10 READ (10,,1,,END=100) RNG,,AZM
IF (ABS (RNG-RNGS) ,GT. 5.) GOTO 11
IF (ABS (AZM-AZMS) ,GT. 10.) GOTO 11
AZMS=AZM
RNGS=RNG
RHOX=RNG*SIN (AZM*PI/180.)/XSCL
RHOY=RNG*COS (AZM*PI/180.)/YSCL
CALL PLOT (XORG+RHOX,,YORG+RHOY,,2)
GOTO 10
C
100 IF (I3G) 101,,101,,52
52 CALL PLOT (0.,0.,-3)
C
51 READ (12,,1,,END=101) RNG,,AZM
RNGS=RNG
AZMS=AZM
RHOX=RNG*SIN (AZM*PI/180.)/XSCL
RHOY=RNG*COS (AZM*PI/180.)/YSCL
CALL PLOT (XORG+RHOX,,YORG+RHOY,,2)
GOTO 51
C
50 READ (12,,1,,END=101) RNG,,AZM
IF (ABS (RNG-RNGS) ,GT. 5.) GOTO 51
IF (ABS (AZM-AZMS) ,GT. 10.) GOTO 51
AZMS=AZM
RNGS=RNG
RHOX=RNG*SIN (AZM*PI/180.)/XSCL
RHOY=RNG*COS (AZM*PI/180.)/YSCL
CALL PLOT (XORG+RHOX,,YORG+RHOY,,2)
GOTO 50
101 CALL PLOT(0.,0.,999)
STOP
1 FORMAT(2(F9.4,1X))
60 FORMAT(1X,'TWO PLOTS? YES/NO')
61 FORMAT(A3)
END