MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A
TECHNICAL REPORT RD-CR-83-16

RANDOM BORESIGHT ERROR ASSESSMENT AND SYSTEMS EVALUATION TEST CHAMBER IMPROVEMENTS

Patrick A. Tilley, Roger G. Gean, and
Timothy A. Palmer
The University of Alabama in Huntsville
School of Engineering
Huntsville, Alabama 35899

April 1983

Prepared for
System Simulation and Development Directorate
US Army Missile Laboratory

U.S. ARMY MISSILE COMMAND
Redstone Arsenal, Alabama 35898

Approved for public release; distribution unlimited.
DISPOSITION INSTRUCTIONS

DESTROY THIS REPORT WHEN IT IS NO LONGER NEEDED. DO NOT RETURN IT TO THE ORIGINATOR.

DISCLAIMER

THE FINDINGS IN THIS REPORT ARE NOT TO BE CONSTRUED AS AN OFFICIAL DEPARTMENT OF THE ARMY POSITION UNLESS SO DESIGNATED BY OTHER AUTHORIZED DOCUMENTS.

TRADE NAMES

USE OF TRADE NAMES OR MANUFACTURERS IN THIS REPORT DOES NOT CONSTITUTE AN OFFICIAL ENDORSEMENT OR APPROVAL OF THE USE OF SUCH COMMERCIAL HARDWARE OR SOFTWARE.
Report Number: RD-CR-83-16

Title: RADOME BORESIGHT ERROR ASSESSMENT AND SYSTEMS EVALUATION TEST CHAMBER UPPROVEMENTS

Authors: Patrick Tilley, Roger Gean, Timothy Palmer

Performing Organization: The University of Alabama in Huntsville

Report Date: April 1983

Security Class: Unclassified

Distribution Statement: Approved for public release; distribution unlimited.

Abstract:
Accomplishments in radome testing and analysis were continued through performance of the task documented by this report. Software improvements for the Radome Positioner and the Radome Measurements Receiver System are presented. The radome testing and post-processing of data are described. Testing and processing difficulties are also discussed.
This technical report was prepared by R. G. Gean, P. A. Tilley, and T. A. Palmer, Electrical and Computer Engineering Department, School of Engineering, The University of Alabama in Huntsville. The purpose of this report is to provide documentation of the technical work performed and of results obtained under delivery order number 0006, contract number DAAH01-82-D-A008; Dr. N. A. Kheir, Principal Investigator. Dr. M. M. Hallum, III, Chief, System Evaluation Branch, Army Missile Laboratory, U.S. Army Missile Command, was technical monitor. Mr. Ernst Evers-Euterneck of the Systems Evaluation Branch provided technical coordination.

The authors wish to acknowledge the valuable discussions and assistance provided throughout the task by L. Ragland of the Systems Evaluation Branch and C. Adams of the Aeroballistic Analysis Branch.

The technical viewpoints, opinions, and conclusions herein are those of the authors and do not imply policies or positions of the U.S. Army Missile Command.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREFACE</td>
<td>1</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>11</td>
</tr>
<tr>
<td>1.0 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2.0 SYSTEM EVALUATION TEST CHAMBER (SETC) PREPARATION</td>
<td>2</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>2</td>
</tr>
<tr>
<td>2.2 Phase Offset</td>
<td>2</td>
</tr>
<tr>
<td>2.3 Software Improvements</td>
<td>2</td>
</tr>
<tr>
<td>2.3.1 Radome Positioner Improvements</td>
<td>2</td>
</tr>
<tr>
<td>2.3.2 Radome Measurements Receiver System Improvements</td>
<td>4</td>
</tr>
<tr>
<td>3.0 SETC TESTING</td>
<td>6</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>6</td>
</tr>
<tr>
<td>3.2 Group 1 - Acceptable Radomes</td>
<td>6</td>
</tr>
<tr>
<td>3.3 Group 2 - Obstructed Radome</td>
<td>6</td>
</tr>
<tr>
<td>3.4 Group 3 - No Radome</td>
<td>6</td>
</tr>
<tr>
<td>4.0 DATA PROCESSING</td>
<td>7</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>7</td>
</tr>
<tr>
<td>4.2 Removing Offset</td>
<td>7</td>
</tr>
<tr>
<td>4.3 Plotting Programs</td>
<td>7</td>
</tr>
<tr>
<td>4.4 Statistical Analysis</td>
<td>7</td>
</tr>
<tr>
<td>5.0 TESTING AND PROCESSING DIFFICULTIES</td>
<td>8</td>
</tr>
<tr>
<td>5.1 Introduction</td>
<td>8</td>
</tr>
<tr>
<td>5.2 Equipment Problems</td>
<td>8</td>
</tr>
<tr>
<td>5.3 Phase Coordinate System</td>
<td>8</td>
</tr>
<tr>
<td>6.0 CONCLUSIONS AND RECOMMENDATIONS</td>
<td>10</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>11</td>
</tr>
<tr>
<td>APPENDICES</td>
<td></td>
</tr>
<tr>
<td>A. Boresight Error Plots</td>
<td></td>
</tr>
<tr>
<td>B. Radome Positioner Software Listing</td>
<td></td>
</tr>
<tr>
<td>C. Radome Measurements Receiver System Software Listing</td>
<td></td>
</tr>
</tbody>
</table>
1.0 INTRODUCTION

The purpose of this report is to review the work performed and the results obtained since completion of the previous delivery order reported in [1]. The goal of this work was to provide a statistical evaluation of IHAWK radomes. The areas of work performed include System Evaluation Test Chamber (SETC) preparations, radome measurements, and data processing.

Section 2 of this report will present the hardware and software changes incorporated to improve the accuracy of data. Changes in test methods, the Radome Positioner software, and the Radome Measurements System software resulted in improved accuracy by improving the acquisition process. The actual tests performed and the results of those tests are found in Section 3 while Section 4 provides a description of the processing means used to analyze and to plot data. Section 5 presents an outline of the major problems and difficulties encountered during performance of the task. Conclusions and recommendations for possible future improvements are presented in Section 6.
2.0 SYSTEM EVALUATION TEST CHAMBER (SETC) PREPARATION

2.1 Introduction

SETC preparations continued throughout this task and included reconfiguring the hardware for each entrance to the SETC. This was essential since several groups share the SETC and each requires a different configuration.

A modification to the test configuration was made by using an offset transmitter horn configuration to induce a known error. The error must stay sufficiently large for the network analyzer to register a stable phase measurement. Another SETC modification consisted of new software written in Motorola 6800 assembly code and Fortran for the Radome Positioner and Radome Measurements Receiver System. This software provides improved operating systems which produce more usable, more complete data sets.

2.2 Phase Offset

Phase offsets measured by the network analyzer provide the signs of error when plotting data. For radomes with very small boresight errors the network analyzer could not provide a stable phase measurement. Thus, a need existed to find a method to accurately test radomes with the available equipment.

Studying the problem led to offsetting the transmitting horn by a known amount. The receiving antenna was boresighted on this offset horn. After boresighting, the transmitting horn was returned to its original center position and scans were executed. Using this method, an inherent error (the amount of offset), which is of a magnitude large enough for the network analyzer to operate accurately, exists. This offset is later removed during data processing to reveal the true magnitude and phase of the boresight error. The arrangement of the transmitting horn is illustrated in Figure 1.

2.3 Software Improvements

Software improvements were made to both the radome positioner operating system and the radome measurements receiver system. Improvements in the positioner are discussed in Section 2.3.1, and those for the measurements receiver are discussed in Section 2.3.2.

2.3.1 Radome Positioner Improvements

A procedure has been developed for easy modification of the Radome Positioner operating system. Working from previous documentation, source files have been reconstructed and stored on master source file diskettes with a Tektronix 8002a microprocessor development system. Using these source files, the positioner program can be modified and recompiled as needed. Subsequently, PROMs can be programmed with modified programs and inserted into the M6800 based microcomputer which controls the positioner.
FIGURE 1 TRANSMITTING HORN ARRANGEMENT
Presently, all modifications are implemented using the Tektronix 8002a. However, the source files could be transferred to any other 6800 Development System.

Using the method described above, a new scan pattern has been implemented on the positioner. The previous positioner patterns were not adequate for thorough analysis of radome boresight error and antenna pattern measurements. This inadequacy existed because the area covered by the scan was not sufficient to yield the full scope of information about the antenna or radome. A careful analysis of the type of data needed revealed that the scan pattern illustrated in Figure 2 was the most useful pattern. The two most pertinent reasons were that (1) the shape of the radome suggested a circular window and (2) the method of data analysis intimated a raster-type scan.

The new pattern covers all data points necessary for current boresight error analysis and antenna pattern measurements. Execution of the pattern (number 3) requires the following key-in sequence:

```
PROG 3
32.0
1.0
```

where 32.0° is the radius of the circular scan region and 1.0 is the number of degrees between each azimuth scan. The revised operating system is presented in Appendix B.

### 2.3.2 Radome Measurements Receiver System Improvements

A procedure has been developed for modification of the Radome Measurement Receiver System operating system. Previously, it was believed that all calculations associated with radome boresight error and antenna pattern measurements could be done during "real-time" operation. However, it was determined that the receiver could not sample each point, perform all of the required calculation, and store the results before the radome reached the next sample point for the density of data desired. All of the calculation routines have therefore been removed from the operating system. This resulted in a more than 60% decrease in the total size of the object code file. In addition to this reduction, the FORTRAN main routine was decreased in size causing another 10% reduction of the total operating system size.

Previously, it was necessary for the operator to interact on a machine level in order to select a sample size. This interaction has been eliminated by enabling the FORTRAN main routine to pass the variable to the assembly routine, CNTL. Modifications have been made to CNTL to enable the receiver to sample on integer degrees within ±.043945°. This is important for precise calculations of boresight error slopes. The revised software is presented in Appendix C.
Figure 2: Raster Scan Pattern
3.0 SETC TESTING

3.1 Introduction

The majority of tests covered by this report are tests which have been repeated using the induced offset described in Section 2.2. The tests fall into three groups: (1) acceptable radomes, (2) obstructed radomes, and (3) no radome.

3.2 GROUP 1 - Acceptable Radomes

This group of radomes include those which have passed final quality assurance inspections. Twelve IHAWK radomes in this category were tested. These tests were repeats of prior tests which did not use the induced offset test method. Vector plots obtained after removal of the known bias show uniformly distributed errors. The largest boresight errors are slightly in excess of one-half degree. The areas of least boresight error occur along the azimuth and elevation axes. Examples of these plots are presented in Figures A-1 through A-5 in Appendix A.

3.3 GROUP 2 - Obstructed Radomes

In an effort to better understand the offset horn configuration, an IHAWK radome was tested using a plexiglass obstruction. The purpose of this test was to observe the effect of the offset method on the large errors caused by placing an obstruction in the radome. Plots show that all errors not affected by the blockage are pointing toward the center of the radome. However, azimuth errors affected by blockage are pointing away from the center. The reason for this deviation may be a change in the polarization of the signal as it passes through the obstruction. Plots of this data are also included in Appendix A in Figures A-6 through A-10.

3.4 GROUP 3 - No Radome

Two tests were performed with no radome attached to the gimbal. The first test was run using the standard horn configurations. The purpose of this test was to check the testing equipment and the anechoic chamber; also, this test served as a comparison for the second test with no radome. The second test was made without a radome but with the horn positioned using the offset bias. This test was then compared with the previous test to detect any inconsistencies between the two configurations. The results of both tests were satisfactory with each showing only negligible errors.
4.0 DATA PROCESSING

4.1 Introduction

This section reviews the methods used to provide analysis of the data obtained during radome tests. The plotting programs and the statistical analysis program are discussed. Additionally, the method used to remove the induced bias are explained.

4.2 Removing Offset

With the configuration illustrated in Figure 1 and the dimensions of the anechoic chamber, there exists an offset of approximately 1.8° in both azimuth and elevation. Since exact positioning of the offset horn is unrealistic, the exact position can only be said to be within ±0.1° of the desired value. The error, looking through the center of the radome, should be zero, and the overall average should also be zero. Thus, the average of all errors can be subtracted from each data point to compensate for small errors in alignment. Since the offset error is simply added to the true error, the total average of all errors is subtracted from each data point to compensate for both small errors in alignment and the ± 0.1° unknown offset error.

The average value is subtracted from each point of raw data with the result being stored in a new file containing the true boresight error for each set of azimuth and elevation angles. Data can then be retrieved by one or more of the plotting programs described in the following section. Plots of the data reflect the true boresight error values.

4.3 Plotting Programs

The two types of plots produced during this task were boresight error vector plots and boresight error three-dimensional plots. Minor adjustments to the existing vector plotting program allowed this program to access the file which contained data with the offset removed. These vector plots display azimuth and elevation errors in the form of a pointing vector emanating from a footpoint which is placed at integer values of azimuth and elevation angles.

Also utilized were two variations of the three-dimensional plotting routines. The first variation was the CARPET program which is normalized and produces a three-dimensional plot with all points connected by lines. The second was the program, CARPET 2, which is the same as CARPET except that it removes the lines which should be hidden, producing an easily readable plot. Examples of these plots are included in Appendix A, and a more detailed description of the actual plotting programs can be found in [1].

4.4 Statistical Analysis

The same methods of statistical analysis described in the report "Radome Boresight Assessment" [1] were utilized during this task. Data collected in the radome tests has been stored to facilitate rapid statistical analysis.
5.0 TESTING AND PROCESSING DIFFICULTIES

5.1 Introduction

Fewer difficulties arose during this task than during previous tasks. However, the following equipment and phase coordinate problems are considered significant.

5.2 Equipment Problems

At the beginning of the testing period, two radomes were tested and analyzed with the results being acceptable. Based on these results, the remaining tests were conducted prior to plotting and analyzing the data. This was required due to time limitations in the SETC. After testing was completed, discrepancies in phase were discovered in the last nine sets of data. This problem was traced to a phase line being grounded. After this problem was corrected, it was necessary to repeat the tests which yielded unacceptable data. Repeated tests provided acceptable results.

Another topic for discussion in this section is the computer system available for radome analysis. Although there has been a significant decrease in down time, terminal access remains a problem. Inavailability of terminals connected to the system has caused several delays.

5.3 Phase Coordinate System

Processing statistical evaluations of several sets of raw data unexpectedly revealed overall boresight averages with a negative azimuth value and with a positive elevation value. Since this was not expected while using the offset method described in Section 2.2, investigations to find the cause were conducted.

The problem was eventually solved by showing that the test equipment is arranged in a manner which logically reverses the apparent error during the test. For an observer positioned behind the antenna and looking in the direction of \( R_a \) (see Figure 3), a positive pitch (elevation) error indicates that the actual target location is above the \( R_a \) axis, as expected. However, a positive yaw (azimuth) error indicates that the actual target location is to the left of \( R_a \) in the yaw plane [2].
Figure 3  Antenna/Radome Geometry for IHAWK Missile
6.0 CONCLUSIONS AND RECOMMENDATIONS

Plots resulting from the use of the new offset horn method are accurate, readable, and present a better overall representation of the true boresight error induced by the radome. Software improvements have proven to be reliable and time-saving tools aiding in the performance of radome testing. Due to these software improvements, the possibility now arises to study boresight error slopes. Boresight error slope is an important characteristic since it indicates the rate of change of the boresight error. Knowledge of this rate is significant for missile response since it can be used to predict future boresight errors by utilizing the information of target movement.

The majority of the recommendations presented in [1] have been implemented. A time-saving recommendation repeated here is the need for a means to transfer data directly from the test system's floppy disks to the data processing system. This modification would also eliminate the need to acquire data cassette tapes. Other recommendations include improved plotting capabilities (possibly on site) and the acquisition of a graphics terminal dedicated to radome testing.
REFERENCES


Appendix A
Boresight Error Plots
Figure A-1
Vector Plot of Boresight Error for Acceptable IHAWK Radome - Case I
(Scale: Distance between dots = $1/2^\circ$ of error = $1^\circ$ of position)
Figure A-4
Carpet II Plot of Azimuth Error for Case I

A-4
Figure A-5
Carpet II Plot of Elevation Error for Case I
Figure A-7

Carpet Plot of Azimuth Error for Case II

A-7
Carpet Plot of Elevation Error for Case II
Figure A-9

Carpet II Plot of Azimuth Error for Case II
C FORTRAN SOURCE DRIVER FOR RMP-V2.1 TIM PALMER

C INITIALIZATION

COMMON ISAMP
ISAMP = 23.0

1 PRINT 2
PRINT 4
PRINT 6
PRINT 8

2 FORMAT (' THE UNIVERSITY OF ALABAMA IN HUNTSVILLE')
4 FORMAT (' RADOME MEASUREMENTS RECEIVER VERSION 2.1')
6 FORMAT (' MODIFIED: 10-82')
8 FORMAT (' BY: T PALMER')
PRINT 10
PRINT 12
PRINT 14

10 FORMAT (' 1) ANTEENA SHOULD BE BORESIGHTED')
12 FORMAT (' 2) POSITIONER GIMBAL SPEEDS SHOULD BE SET')
14 FORMAT (' 3) POSITIONER SHOULD BE AT FIRST SAMPLE POINT')
15 PRINT 16
16 FORMAT (' 1) BORESIGHT ANTEENA; 2) CONTINUE')
READ 18, IDUM

18 FORMAT (50, 20, IDUM)
GO TO 20
GO TO 15

20 PRINT 22
READ 24, SAM
GO TO 20

22 FORMAT (' ENTER SAMPLE SIZE (0.5-2.0):')
READ 24, SAM
IF (SAM .LT. 0.5 .OR. SAM .GT. 2.0) GO TO 20
ISAMP = SAM
24 FORMAT (50, 20, IDUM)
GO TO 22

245 FORMAT ('')
26 FORMAT (' 1) GO, 2) ABORT, E (ANY TIME) = ESCAPE')
CALL OPENIT
25 PRINT 26
READ 28, IDUM
28 FORMAT (50, 20, IDUM)
GO TO 25

30 FORMAT (' TEST NOW ACTIVE')
CALL CHTL
31 PRINT 31
FORMAT (' END OF TEST')
GO TO 1
32 PRINT 34
FORMAT (' TEST ABORTED')
GO TO 1
34 PRINT 52
FORMAT (' REBOOT SYSTEM TYPE LOAD SIG')
END
HAM
CNTL

* OPT REL
* \text{XREF} DSC T YBUF2 A EBIN ELBIN AZP ELP FIRST P TDAZ TDCL
\text{XREF} PSCT NETANL RADLO AIM INIT WRITE CLOSIT
\text{XDEF} CTNL
*
CSCT
*
ISAMP RMB 2
DSCT RMB 2
ABSAZ RMB 2
ABSEL RMB 2
AZCT RMB 2
ELCT RMB 2
LSAZ RMB 2
LSCL RMB 2
*
PSCT
*
CTNL JSR AIM INIT
JSR AIM *GET ANGLES INTO AEBIN ELBIN
CMPI A #05H
BNE SMPL
JMP TSTND
SMPL JSR AIM
LDA A EBIN
STX ABSAZ
LDA E B ELBIN
STX ABSEL
TAD LDA A EBIN
AND A #08H
BEO PAZ
*CHECK FOR NEGATIVE ANGLE
*GO IF POSITIVE
MAZ LDA A #00H
LDA B #00H
SUB A AEBIN+1
SBCB AEBIN
STA A ABSAZ+1
STA A ABSAZ
PAZ LDA A ABSSAZ+1
LDA B ABSSAZ
LDA #0000
STX TDAZ
STX AZCT
JSR AZDLP
LDA A ABSAZ+1
LDA B ABSAZ
SBCB A TDAZ+1
SBCB TDAZ
TST B
BNE EXIT
CMPI A #01
BGT EXIT
LDA A AEBIN+1
LDA E B AEBIN
SUB A LSZA+1
SBCB LSZA
BMI LSGTR
EPA ADZTP
LSGTP LDA A LSZA+1

B-2
INIT

LDA A #OFFH
STA A DDRB
LDA B #04H
STA B CRB
STA A DDRB
LDA A #03H
STA A SPCR1
STA A SRCR1
LDA A #01H
STA A SRCR2
RTS

END
* HAM AIM
* OPT REL
* XDEF AIM
* XREF ANY:DATA,MUX,GAIN,CONVFT,STATUS
* XREF DSC:T,GAIN1,CHAN,RAZIN,ELIN

** SUBROUTINE TO READ THE AZ AND EL ANGLES FROM ANALOG PORTS
** ANGLES ARE ON CHAN 3 FOR AZ AND 4 FOR EL

* PSCT
  AIM
  LDA A GAIN1  *SET A/D GAIN
  STA A GAIN
  CLR CHAN  *CLR CHAN POSITION
  LDA A #02  *LOAD MUX TO 02 TO READ CHAN 3
  STA A MUX
  LDA A CHAN  *INITIALIZE CHAN TO 3
  ADD A #3
  BEACK
  STA A CHAN
  STA A CONVFT  *START CONVERSION PROCESS
  CKSTAT
  LDA A STATUS  *WAIT UNTIL STATUS READY
  BPL CKSTAT
  LDA A CHAN
  CMP A #4  *CHAN 3 HAS BEEN READ READ CHAN 4
  BEO CHAN4
  LD:# DATA  *READ AND STORE AZ ANG
  STA A ZEIN
  LDA A CHAN  *INCREMENT CHAN TO 4
  ADD A #1
  STA A CHAN
  INC MUX  *INCREMENT MUX TO 03 TO READ CHAN 4
  BFA BEAC1  *BRANCH EACH TO READ CHAN 4 THE EL ANG
  CHAN4
  LD:# DATA
  STA EELIN
  RTS
  END

B-5
"NETANL" SUBROUTINE

CONTROLS SAMPLING OF SUN, AZIMUTH, AND ELEVATION DATA. ALSO CONTROLS THE CONVERSION OF THIS DATA INTO DIGITAL FORM.

EXIT: DIGITAL (TWO'S COMPLEMENT) DATA IS STORED IN THE ORDER IT WAS TAKEN UNDER THE FOLLOWING LABELS:

AMPSUM, PHASUM, AMPAZ, PHAAZ, AMPFL, PHAFL

NAME: NETANL
OPC: REL

XREF: ANH: GAIN+ STATUS+ DATA+ CONVRT+ MUX+ DDRB
XREF: DSCT: SAVE+ GAIN+ SWITCH+ WHICH+ FOC+ OFFSET+ SWILOC+ SWIFOS
XREF: CHAN+ TIME+ TWO+ OUTLOC
XREF: PSCT: COMPUT+ WAIT
XDEF: NETANL

PSCT

ETANL

PSH A  * SAVE ACC A
PSH B  * SAVE ACC B
STX SAVE+ * SAVE X REGISTER
LDA A GAIN4  * SET GAIN OF ADC AMP
STA A GAIN

CLR SWITCH  * SET SWITCH MEMORY OUTPUT POINTER TO ZERO
CLR WHICH  * SET SWITCH POSITION POINTER TO ZERO
INC WHICH  * INCREMENT SWITCH POSITION POINTER

LDA A WHICH  * LOAD ACC A WITH SWITCH POSITION
CMP A FOUR  * CHECK IF SWITCH POSITION IS STILL VALID
BEQ RESET  * IF NOT, RESET SWITCH

STA A OFFSET  * STORE OFFSET = WHICH TO COMPUTE NEW ADDRESS
LDX SWILOC  * LOAD X REGISTER WITH ADDRESS TO BE CHANGED
JSR COMPUT  * JUMP TO ROUTINE TO COMPUTE NEW ADDRESS
LDA A @X  * LOAD THE CODE FROM THE COMPUTED ADDRESS IN ACC A
STA A DDRB  * SEND THE CODE TO THE PIA TO SWITCH THE SWITCH

CLP CHAN  * SET CHANNEL POINTER TO ZERO
CLR MUX  * SET ACTUAL CHANNEL TO ZERO
LDX TIME  * LOAD X REGISTER WITH TIME CONSTANT FOR WAIT ROUTINE
JSR WAIT  * GO TO WAIT SUBROUTINE
BPA CHAN  * ONLY DO FOLLOWING SECTION WHEN SETTING DATA OUT
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INC CHANN2</td>
<td>Increment actual channel</td>
</tr>
<tr>
<td>LDA A CHAN</td>
<td>Load ACC A with current channel pointer</td>
</tr>
<tr>
<td>ADD A THO</td>
<td>Increment by two</td>
</tr>
<tr>
<td>STA A CHAN</td>
<td>Store new pointer</td>
</tr>
<tr>
<td>STA A CONVRT</td>
<td>Start conversion process by writing into memory</td>
</tr>
<tr>
<td>LDA A STATUS</td>
<td>Check status until ready</td>
</tr>
<tr>
<td>BPL CKSTAT</td>
<td>When ready continue</td>
</tr>
<tr>
<td>LDA A SWITCH</td>
<td>Load ACC A with switch memory pointer</td>
</tr>
<tr>
<td>ADD A CHAN</td>
<td>Add channel pointer</td>
</tr>
<tr>
<td>STA A OFFSET</td>
<td>This is the offset used to compute the output address</td>
</tr>
<tr>
<td>LDX OUTLOC</td>
<td>Load the X register with the address to be changed</td>
</tr>
<tr>
<td>JSR COMPUT</td>
<td>Jump to the routine to compute the new address</td>
</tr>
<tr>
<td>LDA A DATA</td>
<td>Get first byte of data</td>
</tr>
<tr>
<td>STA A 0X</td>
<td>Store in predetermined position</td>
</tr>
<tr>
<td>INX</td>
<td>Increment output address</td>
</tr>
<tr>
<td>LDA A DATA+1</td>
<td>Get second byte of data</td>
</tr>
<tr>
<td>STA A 0X</td>
<td>Store</td>
</tr>
<tr>
<td>LDA A CHAN</td>
<td>Check if only channel 1 has been done</td>
</tr>
<tr>
<td>BEQ CHANN2</td>
<td>If so, go to channel 2</td>
</tr>
<tr>
<td>LDA A SWITCH</td>
<td>Load ACC A with current switch memory pointer</td>
</tr>
<tr>
<td>ADD A FOUR</td>
<td>Increment by four</td>
</tr>
<tr>
<td>STA A SWITCH</td>
<td>Store new pointer</td>
</tr>
<tr>
<td>BRA NEXT</td>
<td>Redo routine for next switch position</td>
</tr>
<tr>
<td>RESET</td>
<td>Load ACC A with code to turn switch off</td>
</tr>
<tr>
<td>STA A SUPOS</td>
<td>Turn switch off</td>
</tr>
<tr>
<td>LDX SAVE</td>
<td>Restore X register</td>
</tr>
<tr>
<td>PUL B</td>
<td>Restore ACC B</td>
</tr>
<tr>
<td>PUL A</td>
<td>Restore ACC A</td>
</tr>
<tr>
<td>RTS</td>
<td>Return to calling routine</td>
</tr>
<tr>
<td>END</td>
<td></td>
</tr>
</tbody>
</table>
SUBROUTINE "COMPUT" ADDS AN EIGHT BIT NUMBER (OFFSET) TO A SIXTEEN BIT NUMBER (ADDRESS)

ENTRY:
- X REGISTER -- ADDRESS
- OFFSET -- OFFSET

EXIT:
- X REGISTER -- MODIFIED ADDRESS
- A AND B ACCUMULATORS ARE SAVED

COMPUT STX LOC  * TEMPORARILY STORE ADDRESS TO BE CHANGED
PSH A  * SAVE ACC A
PSH B  * SAVE ACC B
LDA A LOC  * LOAD ACC A WITH MS BYTE OF ADDRESS TO BE CHANGED
LDA B LOC+1  * LOAD ACC B WITH LS BYTE OF ADDRESS TO BE CHANGED
ADD B OFFSET  * ADD OFFSET TO LS BYTE
ADC A ZERO  * CARRY IF NECESSARY
STA A LOC  * TEMPORARILY STORE NEW ADDRESS
STA B LOC+1  *
LDR LOC  * LOAD THE X REGISTER WITH THE NEW ADDRESS
PUL B  * RESTORE ACC B
PUL A  * RESTORE ACC A
RTS  * RETURN TO CALLING PROGRAM
END
NAM WAIT
OPT REL
XDEF WAIT
PSCT

"WAIT" SUBROUTINE
STALLS <TIME> X (100 MICROSECONDS) WHERE TIME = CONTENTS OF X REGISTER
ACCUMULATORS A AND B ARE NOT AFFECTED

WAIT PSH A * SAVE CONTENTS OF ACC A
* LDA A #00H * LOAD ACC B WITH INITIAL COUNTDOWN
* WAIT1 DEC A * DECREMENT FOR COUNTDOWN
BNE WAIT1 * REDO IF COUNTDOWN IS NOT FINISHED
* LDA A #0FH * LOAD ACC A WITH SUBSEQUENT COUNTDOWN
* DEX **DECREMENT TIME
BNE WAIT1 * USE MORE TIME IF TIME > 0
* CMP A 0:5 * STALL FOR 5 MICROSECONDS
* PUL A * RESTORE CONTENTS OF ACC A
* RTS * RETURN TO CALLING ROUTINE
* END

B-9
NAME: RADLO

OPT: REL

XREF: DSC1:HEADER, FINAL
XREF: DSC1:G1.BSE2.BSEEL.ZBIN.ELEH
XREF: DSC1:AMPSUM.PHASUM.AMPS2.PHASE.AMPEL.PHIPEL
XREF: DSC1:BUFF2
XREF: ANY:GIMSEP.GLSEP.BNMSBP.BMNSBP.BMNSBP.BELEP.BBEL
XREF: ANY:ASSBP.RBCBP.PCMSBP.PSCBP.PBCBP
XREF: ANY:AMISBP.APULBP.PALEBP.FALEBP
XREF: ANY:AREISP.BPSBP.BPISP.FEPSP.FEPSP
XREF: ANY:DRAC,.DARAS.DARAS.DARAS.DARAS
XREF: RADLO

PSCT

"RADLO" (1) LOADS THE RADOME MEASUREMENTS INTO THE BUFFER FOR THE RFSS INTERFACE
(2) LOADS THE DAC'S

ADLO

LDX #6BUF2
STX TxBUF2
STX %6BUF2+24
LDA #G1
LDA B G1+1
STA A 2X
STA B 3X
LDA B BSE2
LDA B BSE2+1
STA A 4X
STA B 5X
LDA A BSEEL
LDA B BSEEL+1
STA A 6X
STA B 7X
LDA A ZBIN
LDA B ZBIN+1
STA A 8X
STA B 9X
LDA A ELEH
LDA B ELEH+1
STA A .10X
STA B 11+X
LDA A AMFSUM
LDA B AMFSUM+1
STA A 12+X
STA B 13+X
LDA A PHASUM
LDA B PHASUM+1
STA A 14+X
STA B 15+X
LDA A AMFAZ
LDA B AMFAZ+1
STA A 16+X
STA B 17+X
LDA A PHAAZ
LDA B PHAAZ+1
STA A 18+X
STA B 19+X
LDA A AMPER
LDA B AMPER+1
STA A 20+X
STA B 21+X
LDA A PHREL
LDA B PHREL+1
STA A 22+X
STA B 23+X

* * *

LDX cl
STX DACAL1
LDX cl
STX DACAL2
LDX cl
STX DACAL3
LDX cl
STX DACAL4
LDX cl
STX DACAL5

* * *

RTS

* * *

END
### DATA

**OPT REL**

<table>
<thead>
<tr>
<th>NAME</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEG</td>
<td>FDB 768EH</td>
</tr>
<tr>
<td>TEN</td>
<td>FDB 0FAB3H</td>
</tr>
<tr>
<td>THENT</td>
<td>FDB 30</td>
</tr>
<tr>
<td>CONI</td>
<td>FDB 00180H</td>
</tr>
<tr>
<td>CON2</td>
<td>FDB 00000H</td>
</tr>
<tr>
<td>DEG2</td>
<td>FDB 058EH</td>
</tr>
<tr>
<td>SETS</td>
<td>FME 1</td>
</tr>
<tr>
<td>TME1</td>
<td>FME 2</td>
</tr>
<tr>
<td>TME2</td>
<td>FME 3</td>
</tr>
<tr>
<td>DEG</td>
<td>FDB 00000H</td>
</tr>
<tr>
<td>30</td>
<td>FME 4</td>
</tr>
<tr>
<td>EL</td>
<td>FDB 00000H</td>
</tr>
<tr>
<td>FAC</td>
<td>FDB 00130H</td>
</tr>
<tr>
<td>FDE 00000H</td>
<td>FDB 005A0H</td>
</tr>
<tr>
<td>FDE 00000H</td>
<td>FDB 00000H</td>
</tr>
<tr>
<td>FDE 00000H</td>
<td>FDB 00000H</td>
</tr>
<tr>
<td>FDE 00000H</td>
<td>FDB 00000H</td>
</tr>
<tr>
<td>G1</td>
<td>RMB 2</td>
</tr>
<tr>
<td>BEHZ</td>
<td>RMB 2</td>
</tr>
<tr>
<td>ESEL</td>
<td>RMB 2</td>
</tr>
<tr>
<td>REFSUM</td>
<td>FDB 0440H</td>
</tr>
<tr>
<td>SUDE</td>
<td>FME 4</td>
</tr>
<tr>
<td>ACBE</td>
<td>FME 4</td>
</tr>
<tr>
<td>ELDB</td>
<td>FME 4</td>
</tr>
<tr>
<td>FIFSTP</td>
<td>FME 1</td>
</tr>
<tr>
<td>R2P</td>
<td>FME 2</td>
</tr>
<tr>
<td>ELG</td>
<td>FME 2</td>
</tr>
<tr>
<td>TRAC</td>
<td>FME 3</td>
</tr>
<tr>
<td>IDDE</td>
<td>FME 2</td>
</tr>
<tr>
<td>ICAIP T</td>
<td>FME 1</td>
</tr>
<tr>
<td>NEMP</td>
<td>FLE 5</td>
</tr>
<tr>
<td>ELP</td>
<td>FCE 6</td>
</tr>
<tr>
<td>NHPAREN</td>
<td>FDB 0FFEH</td>
</tr>
</tbody>
</table>

### NOTES

B-12
FINAL FDB OFF2DH
GDMSFB FCB 1
GDLSEF FCB 2
KAMSFB FCB 3
KALSBP FCB 4
KEMSFB FCB 5
KELSFB FCB 7
SDMSFB FCB 12
SDLSFB FCB 2
ADMSFB FCB 3
ADLSFB FCB 4
EDMSFB FCB 6
EDLSFB FCB 7
GINSEF FCB 2
GILSEF FCB 3
BANSEF FCB 4
BALSFB FCB 5
BENSFB FCB 6
BELSEF FCB 7
ABNSFB FCB 8
ABLSFB FCB 9
EBNSFB FCB 10
EBLSFB FCB 11
ASMSFB FCB 12
ASLSFB FCB 13
PSNSFB FCB 14
PSLSFB FCB 15
AMMSFB FCB 16
AMLSEF FCB 17
PMNSFB FCB 18
PALSFB FCB 19
REMSEF FCB 20
RELSFB FCB 21
PEMSFB FCB 22
PELSFB FCB 23

SFIA FDB 006A3H
SFIP FDB 0138FH
SFIP FDB 00403H
SFG1 FDB 00SCH
SFBS FDB 04000H
THPSIN FDB 00E00H
THPSIN PMB 4

* ASCT
** DAC LOCATIONS
DAC1 EOU 0E02EH
DAC2 EOU 0E02CH
DAC3 EOU 0E03EH
DAC4 EOU 0E03FH
DAC5 EOU 0E03AH

END
* NAM RDATA
* OPT REL
* XDEF TEMPA, TEMFB, TEMPX, BCDSIGN, EINCREMENT, OUTBUF
XDEF AZBCD, AZBCD, ELBCD, ELBCD, AZBIN, AZBIN, TMBPCD
XDEF TEMEUR, T:EUFR, T:EUFR, T:EUFR, T:EUFR, T:EUFR
XDEF AZANG, ELANG, AZANG, PLUS, APST, DCML, CF, HINTP, TXDRE
XDEF SRCR1, RTDP1, SRCR2, RTDP2
XDEF APUDAT, AFUSTA, DDFA1

* DSCT
PLUS FCB 2EH
APST FCB 27H
DCML FCB 2EH
CR FCB 60H
TXDRE FCB 02H

** ALLOCATED MEMORY AREA.
** TEMPORARY MEMORY VARIABLES.
** TEMPA RMB 1 *ACCUMULATOR A (BCDSIGN, EINCREMENT).
TEMFB RMB 1 *ACCUMULATOR B (BINANG).
TEMPX RMB 2 *INDEX REGISTER (INCREMENT).
BCDSIGN RMB 1 *ASCII SIGN OF Elevation BCD Angle (BCDSIGN) ANGLE.
BINANG RMB 1 *USED IN FINDING MS BIT OF BINARY VALUE (BCBIN).
OUTBUF RMB 2 *POINTER TO TEMPORARY OUTPUT BUFFER (INCREMENT).

* DSCT
** DEDICATED MEMORY VARIABLES.
** AZBCD RMB 1 *ASCII SIGN OF Elevation BCD Angle (ANGLE).
AZBCD RMB 2 *PACKED BCD FORM OF AZIMUTH ANGLE (ANGLE).
ELBCD RMB 1 *ASCII SIGN OF Elevation BCD Angle (ANGLE).
ELBCD RMB 2 *PACKED BCD FORM OF Elevation ANGLE (ANGLE).
AZBIN RMB 2 *TWO'S COMPLEMENT OF AZIMUTH ANGLE (ANGLE).
ELBIN RMB 2 *TWO'S COMPLEMENT OF Elevation ANGLE (ANGLE).
TMBPCD RMB 1 *USED IN PACKING Elevation ANGLE (BCDP).
RCURF RMB 2 *POINTER FOR ASCII CHAR IN CURR BUFFER 1 (RCURF).
TXBUFP RMB 2 *POINTER FOR ASCII CHAR IN TXMT BUFFER 2 (TXBUFP).

** EXTERNAL INTERFACE BUFFER AREA

B-14
THFBUF EQU .30  *TEMPORARY BUFFER AREA (BCDANG)*
AZANG EQU THFBUF + LOCATION IN RBUF OF AZIMUTH BCD ANGLE (ANGLE)
ELANG EQU THFBUF+10 + LOCATION IN RBUF OF ELEVATION BCD ANGLE (ANGLE)
RBUF FDB 00000H  +ASCII CHAR READ FROM POSITIONER (BCDANG+RCVRPP).
FDB 00000H
FDB 00000H
FDB 00000H
FDB 00000H
FDB 00000H
FDB 00000H
FDB 00000H
TXBUF1 RMB 30
TXBUF2 RMB 26

* * *
ASCT

** I/O DEVICE EQUATE SYMBOLS

* *
SRCR1 EQU 0E004H
RTDR1 EQU 0E005H
SRCR2 EQU 0E006H
RTDR2 EQU 0E007H

* * * APU DATA

* APUAD1 EQU 0E00CH
APUSTA EQU 0E00DH
DDRA1 EQU 0E00EH
AINTP EQU 0FFF8H

END
HAM  DDATA

OPT  REL

XDEF  SAVEX:LOC SWITCH:WHICH:CHAN:OFFSET
XDEF  SWILOC:SHIPOS:CRA
XDEF  GAIN1:GAIN2:GAIN3:GAINS
XDEF  TIME:ZERO:THO:THO

DSCT
GAIN1  FCB  00H
GAIN2  FCB  01H
GAIN3  FCB  02H
GAINS  FCB  03H
TIME   FDB  0002H
ZERO   FCB  00H
THO    FCB  02H
THO    FCB  04H

** ALLOCATED MEMORY AREA

** TEMPORARY MEMORY VARIABLES

SAVE:  RME  2  * USED TO SAVE CONTENTS OF REGISTERS
LOC:   RME  3  * USED WHEN COMPUTING ADDRESS PLUS OFFSET
SWITCH RME  1  * USED IN INCREMENTS PAGE OFFSET MEMORY POSITION
   (SOFTWARE SWITCH
WHICH  RME  1  * POINTER USED TO KEEP TRAIC OF SWITCH POSITION
CHAN   RME  1  * POINTER USED TO KEEP TRACI OF CHANNEL SELECTION
OFFSET RME  1  * COMPUTED OFFSET USED WHEN COMPUTING NEW ADDRESS

** DEDICATED MEMORY VARIABLES

B-16
SWILOC FDB SWIPOS * LOCATION OF SWITCH POSITION CODES
SWIPOS FCB OFFH * SWITCH POSITION -- OFF
FCB OFFH * SWITCH POSITION -- ONE
FCB OFFH * SWITCH POSITION -- TWO
FCB OFFH * SWITCH POSITION -- THREE
FCB OFFH * SWITCH POSITION -- FOUR

** EXTERNAL INTERFACE BUFFER AREA

OUTLOC FDB AMPSUM * LOCATION OF MEMORY RESERVED FOR OUTPUT
AMPSUM FDB 0001H * SUM AMPLITUDE
PHASUM FDB 0001H * SUM PHASE
AMPL FDB 0001H * AZIMUTH AMPLITUDE
PHAS FDB 0001H * AZIMUTH PHASE
AMPHEL FDB 0001H * ELEVATION AMPLITUDE
PHAE FDB 0001H * ELEVATION PHASE

** EXTERNAL INTERFACE BUFFER AREA

CRA RMB 2 * TERMINATES CONTROL BASED ON AN "S"
ASCT EOU 0E002H * DATA DIRECTION REGISTER PIA PORT B
(SWITCH CONTROL)
CPB EOU 0E003H * CONTROL REGISTER PIA PORT B
BASE EOU 0E004H * BEGINNING OF ADC BOARD "MEMORY" LOCATIONS
GAIN EOU BASE+0H * MEMORY LOCATION TO ACCESS GAIN
MUX EOU BASE+0AH * MEMORY LOCATION TO ACCESS MULTIPLE CHANNEL SELECT
CONVRT EOU BASE+0CH * MEMORY LOCATION TO ACCESS CONVERT COMMAND
STATUS EOU BASE+OCH * MEMORY LOCATION TO ACCESS STATUS
DATA EOU BASE+ODH * MEMORY LOCATION TO ACCESS ADC DATA MS BYTE

END
NO UNDEFINED SYMBOLS

MEMORY MAP

<table>
<thead>
<tr>
<th>Section</th>
<th>Start</th>
<th>Size</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>0000</td>
<td>0040</td>
<td>0000</td>
</tr>
<tr>
<td>C</td>
<td>0002</td>
<td>2000</td>
<td>0002</td>
</tr>
<tr>
<td>D</td>
<td>0035E</td>
<td>2002</td>
<td>2235F</td>
</tr>
<tr>
<td>P</td>
<td>1857</td>
<td>2360</td>
<td>3E66</td>
</tr>
</tbody>
</table>

MODULE NAME: BECT DECT PSCT

<table>
<thead>
<tr>
<th>Section</th>
<th>Start</th>
<th>Size</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td>0040</td>
<td>2002</td>
<td>2360</td>
</tr>
<tr>
<td>CNTL</td>
<td>0040</td>
<td>2002</td>
<td>274E</td>
</tr>
<tr>
<td>INIT</td>
<td>0040</td>
<td>2002</td>
<td>2874</td>
</tr>
<tr>
<td>AIM</td>
<td>0040</td>
<td>2002</td>
<td>2884</td>
</tr>
<tr>
<td>NETAIL</td>
<td>0040</td>
<td>2002</td>
<td>2884</td>
</tr>
<tr>
<td>RADLO</td>
<td>0040</td>
<td>2002</td>
<td>2936</td>
</tr>
<tr>
<td>COMPUT</td>
<td>0040</td>
<td>2002</td>
<td>2956</td>
</tr>
<tr>
<td>WAIT</td>
<td>0040</td>
<td>2002</td>
<td>29F0</td>
</tr>
<tr>
<td>SDATA</td>
<td>0040</td>
<td>2002</td>
<td>3A00</td>
</tr>
<tr>
<td>RDATA</td>
<td>0040</td>
<td>2002</td>
<td>3A00</td>
</tr>
<tr>
<td>DDATA</td>
<td>0040</td>
<td>2102</td>
<td>3A00</td>
</tr>
<tr>
<td>FTHRUN</td>
<td>0040</td>
<td>2135</td>
<td>3C38</td>
</tr>
</tbody>
</table>

COMMON SECTIONS

<table>
<thead>
<tr>
<th>Name</th>
<th>Start</th>
<th>Size</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>S</td>
<td>SIZE</td>
<td>STR</td>
</tr>
<tr>
<td>ADDR</td>
<td>P</td>
<td>0000</td>
<td>3EA8</td>
</tr>
<tr>
<td>.ADRDC</td>
<td>D</td>
<td>0002</td>
<td>235E</td>
</tr>
</tbody>
</table>

DEFINED SYMBOLS

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td>P 2360</td>
</tr>
<tr>
<td>CNTL</td>
<td>P 274E</td>
</tr>
<tr>
<td>INIT</td>
<td>P 2850</td>
</tr>
<tr>
<td>AIM</td>
<td>P 2874</td>
</tr>
<tr>
<td>NETAIL</td>
<td>P 2884</td>
</tr>
<tr>
<td>RADLO</td>
<td>P 2936</td>
</tr>
<tr>
<td>COMPUT</td>
<td>P 2956</td>
</tr>
<tr>
<td>WAIT</td>
<td>P 29F0</td>
</tr>
</tbody>
</table>

...
RLOAD
BASE
IDON
LOAD=LIB
LIB=FORLB
MO=MAP1.MO
MAPF
MO=CH
MAPF
OBJA=OBJ1.LO
LOAD=LIB
LIB=FORLB
EXIT

MERGE FSTP.R0:TPCHT.R0:INIT.R0:1:AIM.R0:1:COMPUT.R0:1:TODISK.R0:1:LIE.R0
MERGE LIB.R0:NETAIL.R0:1:RADLO.R0:1:WAIT.R0:1:DATA.R0:1:LIE.R0
Appendix C
Radome Measurements Receiver System Software Listing
NEW SCAN PATTERN (#3) SCANS A 32 DEGREE CIRCULAR RASTER PATTERN

ENTRY 1: MUST BE 32.0 OR WHATEVER ELEVATION X -1 TO STOP AT
ENTRY 2: ELEVATION STEP SIZE
ENTRY 3: ANY KEY TO START SCAN

ST28: LDX #0000
STX MFLAG ;CLEAR MOTOR AND PROGRAM FLAGS
STX AZIEY ;FIRST AZIMUTH
LDX #3200H ;RADIUS
STX ELKEY ;NOW ELEVATION
LDX #28BH ;"",""
STX AZIEYS
LDX #ST28
STX STADDR ;SAVE RETURN ADDRESS
DEC FFLAG ;SET PROG FLAG
NOP ST280
NOP
NOP
ST28A LDA A ELSIGN ;GET CURRENT ELEVATION SIGN
CMF A #2BH ;""
BLX ST28C ;GO IF ELEVATION <0
LDA A ELIEOD ;B2D CURRENT ELEVATION
LDA B ELIEOD+1
LDX #FROGA ;LAST ELEVATION MAGNITUDE
NOP
CMP A #1 ;CHECK PROXIMITY OF LAST POINT
BMI ST28AA ;NOT WITHIN 1 DEGREE
BRA FINA ;WITHIN ONE DEGREE, DONE ST28AA TST CARRY ;MAYBE OVER 1 DEGREE PAST END
BMI ST28B ;NO CONTINUE FINA
LDX #ST28
STX STADDR
JMP ST28
ST28B LDA A AZSIGN ;CALCULATE NEW AZIMUTH
STA A AZIEYS
LDA #ST28D1
STX STADDR
NOP
ST28B1 LDA A ELKEY+1
ADD A #FROGA+1
DAA
STA A ELKEY+1
LDA A ELKEY
ADC A FROGB
DAA
STA A ELKEY
BRA ST28A
ST28B2 LDX #ST28B3
STX STADDR
NOP
ST28B3 LDA A AZSIGN ;CURRENT AZ SIGN
CMF A #2BH ;""
LDA #28BH
LJR A #30H ;""
STA A AZIEYS
BRA ST28B4
ST28B4 LDA A AZIEYS ;CHANGES SIGN OF AZIMUTH FOR NEXT POSITION
ST28B5 LDX #10DA
DAA

C-1
JMP ST28A
LDA A ELKEY
LDA B ELKEY+1
LDX #PROGB
NOP
STA A ELKEY
STA B ELKEY+1
TST CARRY
BEO ST28C1
LDA A #2DH
STA A ELKEY
STX #ST28C2
STX STADDR
NOP
ST28C2 JSR NEWAZ
LDA A AZSIGN
STA A AZEYS
LDX #ST28C3
STX STADDR
NOP
ST28C3 LDA A AZSIGN
CMP A #26H
BEO ST28C4
LDA A #26H
STA A AZEYS
ERA ST28C5
ST28C4 LDA A #2DH
STA A AZEYS
ST:28C5 LDX #ST28A
STX STADDR
JMP ST28A

"NEWAZ" SUBROUTINE THAT CALCULATES AZ POSITION BASED ON EL. AND PROGB

NEWAZ LDA A ELKEY
JSR SC00IN
LDA A TEMPA
ADD A TEMPA
STA A SAVEA
LDY #TAB1
STX SAVEX1
LDA A SAVEX1+1
LDA B SAVEX1
ADD A SAVEA
ADC B #00H
STA A SAVEX1+1
STA B SAVEX1
LDX SAVEX1
LDA A 0.X
STA B 1.X
STA A AZKEY
STA B AZKEY+1
RTS

TAB1 WORD 02300H
WORD 02198H
WORD 03194H
WORD 03186H
WORD 03175H
WORD 03161H
WORD 03143H
WORD 02122H
WORD 02100H
WORD 03070H
WORD 03040H
WORD 02000H
WORD 02046H
WORD 02724H
PCDBIN CLR TEMPA
PSH A #0F0H ; SAVE A
AND A TEMPA  ; GET UPPEP 4 BITS
LSR A ; MOVE 1 BIT RIGHT
LSR A ; THAT MAKES 2
LSR A ; THERES 3
LSR A ; THAT'S ALL FOLKS
CLC ; CLEAR ANY CARRY

TNLP TST A
SEC ONELP  ; SAVE SHIFTED A (STACK: SHA,A)
CLR TEMPA
PSH A

LP1 LDA A TEMPA  ; SAVE SHIFTED A BACK
ADD A #0AH  ; ADD 10
STA A TEMPA  ; ADD COUNTER
FUL A  ; GET SHIFTED A AGAIN
DEC A ; IF FINISHED
BEC ONELP

ONELP FUL A  ; FULLS ORIGINAL A
AND A #0FH  ; GET LS NIBBLE
TST A  ; SEE IF DONE
PSU EBBC  ; GO IF NO ONES
PSH A

LP2 LDA A TEMPA  ; GET INTERMEDIATE RESULT
ADD A #01H  ; INCREMENT BY ONE
STA A TEMPA  ; SAVE RESULT
FUL A  ; GET ONE COUNTER
DEC A ; DECREMENT
BEC EBBC  ; DO IF DONE
PSH A  ; NOT DONE YET

EBBC  ; NOT DONE YET

END
<table>
<thead>
<tr>
<th>SECTION</th>
<th>ORG</th>
<th>0000H</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISBUF</td>
<td>BLOCK</td>
<td>20</td>
</tr>
<tr>
<td>SIGUF</td>
<td>BLOCK</td>
<td>21</td>
</tr>
<tr>
<td>TEMFX</td>
<td>BLOCK</td>
<td>2</td>
</tr>
<tr>
<td>RAIIUB</td>
<td>BLOCK</td>
<td>2</td>
</tr>
<tr>
<td>FPT32</td>
<td>BLOCK</td>
<td>4</td>
</tr>
<tr>
<td>ELRESULT</td>
<td>BLOCK</td>
<td>5</td>
</tr>
<tr>
<td>AZRESULT</td>
<td>BLOCK</td>
<td>5</td>
</tr>
<tr>
<td>INTREG</td>
<td>BLOCK</td>
<td>2</td>
</tr>
<tr>
<td>ENTRY1</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>ENTRY2</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>SAVEB</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>SAVEC</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>CHARBF</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>CHARPT</td>
<td>BLOCK</td>
<td>2</td>
</tr>
<tr>
<td>CHARCT</td>
<td>BLOCK</td>
<td>2</td>
</tr>
<tr>
<td>CHRNUM</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>SAVEA</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>SAVEV</td>
<td>BLOCK</td>
<td>2</td>
</tr>
<tr>
<td>SAVEX1</td>
<td>BLOCK</td>
<td>2</td>
</tr>
<tr>
<td>TEMPA</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>TEMPB</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>MSFENC</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>LUSENC</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>LETA</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>LETE</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>BCDRA</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>ECDR</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>SAVEDEC</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>ANGLE</td>
<td>BLOCK</td>
<td>3</td>
</tr>
<tr>
<td>SIGN</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>ASIGN</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>ELSIGN</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>*AZBCD</td>
<td>BLOCK</td>
<td>2</td>
</tr>
<tr>
<td>AZBCD</td>
<td>BLOCK</td>
<td>2</td>
</tr>
<tr>
<td>TEMPX1</td>
<td>BLOCK</td>
<td>2</td>
</tr>
<tr>
<td>TEMFX2</td>
<td>BLOCK</td>
<td>2</td>
</tr>
<tr>
<td>ENTRYX</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>ENTRYB</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>EVENT</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>TEMPA1</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>TEMPB1</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>EYC</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>AZKEY</td>
<td>BLOCK</td>
<td>2</td>
</tr>
<tr>
<td>ELKEY</td>
<td>BLOCK</td>
<td>2</td>
</tr>
<tr>
<td>ADKEYS</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>ELKEYS</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>MFLAG</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>PFLAG</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>MINGEN</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>SPEEDA</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>SPEEDE</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>AZMAG</td>
<td>BLOCK</td>
<td>2</td>
</tr>
<tr>
<td>ELMAG</td>
<td>BLOCK</td>
<td>2</td>
</tr>
<tr>
<td>ATEL</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>TEMPS</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>PROGN</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>PROGL</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>PRODA</td>
<td>BLOCK</td>
<td>2</td>
</tr>
<tr>
<td>PRODB</td>
<td>BLOCK</td>
<td>2</td>
</tr>
<tr>
<td>PROGC</td>
<td>BLOCK</td>
<td>2</td>
</tr>
<tr>
<td>FLAG</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>PLAGAE</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>LPLAGAE</td>
<td>BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>SFLAGA</td>
<td>BLOCK</td>
<td>1</td>
</tr>
</tbody>
</table>

C-4
BFLAG BLOCK 1
TEMPD BLOCK 1
BCDSVR BLOCK 1
FPTEL BLOCK 1
FPTAZ BLOCK 1
FPTELB BLOCK 1
PROCNT BLOCK 1
STATER BLOCK 2
PADMRG BLOCK 2
BINANG BLOCK 1
SINE BLOCK 1
COSINE BLOCK 1
SIGN BLOCK 1
CSIGN BLOCK 1
SAVE1 BLOCK 1
BINUPR BLOCK 2
CELLIM BLOCK 1
PELLIM BLOCK 2
PBLA2 BLOCK 2
NAZLIM BLOCK 2
LFLAGE BLOCK 1
LFLAGA BLOCK 1
MSGFLG BLOCK 1
SAVES2 BLOCK 2
MODIFICATION V1.3

I/O EQUATES

*AFSPD EQU 00E0H
*AFSPD EQU 00E7H
DFFA2 EQU 00404H ;MS 4 BITS OF DAC #1 - AZIMUTH
CFA2 EQU 00406H ;LS 8 BITS OF DAC #1 - AZIMUTH
CRB2 EQU 00407H
DFFA3 EQU 00800H ;LS 4 BITS OF DAC #2 - ELEVATION
CFA3 EQU 00801H
DFFB3 EQU 00802H ;MS 8 BITS OF DAC #2 - ELEVATION
CBA2 EQU 00803H
DISA2 EQU 0000H
DISEL EQU 0000AH
MSBSEL EQU 00807H
LSBSEL EQU 0080CH
MSBGA2 EQU 00E01H
LSBGAR EQU 00E00H
DRRA EQU 00800H
CRA EQU 00801H
DDBA EQU 00802H
CRA EQU 00803H
ACIAS EQU 00804H ;ACIA STATUS/CONTROL REGISTER
ACIID EQU 00808H ;ACIA DATA REGISTER
APUDATA EQU 00804H ;AIFU DATA INPUT/OUTPUT
APUDATR EQU 00800H ;AIFU COMMAND INPUT AND STATUS OUTPUT
ETHSPD EQU 0F0H ;(MOD 1.3)

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

II II
MAIN PROGRAM- 2000-2016 HEX

II II

C-5
SECTION MAIN
ORG 2000H

;INITIALIZE PIAS

GDE LDS #0FFFH
NOP
SEI
LDA A #$3 ;00000011 = MASTER RESET
STA A ACIAS ;RESET ACIA
LDA A #$B1H ;10000001 = 7 BITS, EVEN PARITY, 2 STOP BITS
STA A ACIAS ;SET ACIA FOR RECEIVER INT,TXMIT INT OFF
CLR CRA ;CLEARS CONTROL REG A
CLR CRB ;CLEARS CONTROL REG B
CLR CRA2
CLR CRB2

;ADDITION TO INIT ROUTINE

;COMPENSATION FOR INVERTED BUS DRIVERS
; (MOD 1.2)

LDA A #0FFH ;GET OPPOSITE OF 00
STA A CRA3 ;CLEAR CONTROL REG OF A SIDE
STA A CRB3 ;CLEAR CONTROL REG OF B SIDE
CLR DDR3 ;SETS UP B SIDE OF DATA PORT A ALL OUTPUTS
LDX #8400H
LDA A #$FOH
STA X
STA A DDOA3 ;CLEAR CONTROL REG OF A SIDE
STA A CRB3 ;CLEAR CONTROL REG OF B SIDE
CLR DDR3 ;SETS UP B SIDE OF DATA PORT A ALL OUTPUTS

;INITIALIZED MOTORS TO ZERO SPEED, ETC.

LDA A #39H ;ASCII "9"
STA A AZSPEED ;AZ GIMBAL AT FULL SPEED
STA A ELSPEED ;ELEVATION GIMBAL AT FULL SPEED
LDX #7500H
STX FCELLIM
INITIALIZES CONTROL LOOP SUCH THAT THE POSTIONER
WILL NOT MOVE UPON POWER-UP (MOD 1.1)

JSR SHAENC ;READ ANGLES
LDX AZBCD
STX AZEY ;UPDATES AZIMUTH KEYENTRY WITH
LDX ELBCD ;CURRENT AZIMUTH LOCATION
STX EL EY ;SAME FOR ELEVATION
STX AZSIGN ;CURRENT AZ SIGN STATUS
NOP ;FIX FOR THE "CLI" INSTR THAT FOLLOWS

CLI ! "BEGIN STATE TABLES"

MSGA LDX #MSG12 JSR ASCDIS ;DISPLAY "THE GA. TECH RFSS"
LDA B #20 JSK WAIT FOR 2 SECONDS
LDA #MSG12 JSR ASCDIS ;DISPLAY "RADOME POSITIONER"
LDA B #10 JSK WAIT FOR 1 SECOND
LDA #MSG14 JSR ASCDIS ;DISPLAY "VERSION 1.5"

MSGB LDA B #10 JSR WAIT

STATE ZERO MAIN CONTROL LOOP
LDA A #FLAG ;IDLE STATE
BFL STOA
CLR #FLAG ;CLEARS "KEY PRESSED" FLAG
LDA A KEYENT ;GETS KEYCODE OF KEY PRESSED
LDX #PO ;PUTS 0 STATE POINTER IN INDEX REG
JSR ADDCAL ;JUMPS TO ROUTINE TO CALC NEXT STATE
LDX 0,X JMP 0,X

AZIMUTH MOTOR CONTROL LOOP
JSR SHAENC ;READS AZ AND EL ANGLES
LEA A MFLAG
BMI STOA
LEA A AZEYS
BMI LEA A AZSIGN ;SKE IF BOTH SIGNS EQUAL
LDA A AZKEY+ ;ADD A AZBCD+1
STA A AZMAG+1
DAA
LDA A AZKEY
DAA
STA A AZMAG
BRA STOX2
LDA A AZKEY+1
LDA #AZBCD
JSR BCDSUB ;JUMPS TO ROUTINE TO SUBTRACT BCD #S
STA A AZMAG+1
STA B AZMAG+1

;ADDITION TO TIGHTEN CONTROL LOOP (AZ) TO .1 DEGREE (MOD 1.1) ;
STOX2 TST A ;START <0.2 DEGREE TEST
BNE STOX1 ;BRANCHES TO <0.5 DEG TEST IF BCD WORD NOT <0.2
CMP B #15H ;COMPARING TO 0.15 DEG
BHI STOX1 ;BRANCHES TO <0.5 DEG TEST IF BCD WORD NOT <0.2
LDA A #THSPD ;SET SPEED TO EIGHTH SPEED
STA A SPEEDA ;SET UP SPEED VARIABLE FOR USE LATER
BRA STOD

STOX1 TST A ;START <0.5 DEG TEST
BNE STOB ;BRANCHES IF <0.5 DEG TEST IF BCD WORD NOT <0.5
CMP B #00H ;TEST FOR <5 DEG
BHI STOX1 ;BRANCHES TO <0.5 DEG TEST IF BCD WORD NOT <0.2
LDA A #QUASPD ;SET SPEED TO QUARTER SPEED
STA A SPEEDA ;USED LATER
BRA STOC

STOB CMP A #00H ;TEST FOR <5 DEG
BHI STOC ;BRANCH TO <10 DEG IS <5 DEG TEST FAILS
LDA A #QUASPD ;SET SPEED TO QUARTER SPEED
STA A SPEEDA ;USED LATER
BRA STOC

STOC LDA A AZSPD ;SET SPEED TO USER SELECTED SPEED (MOD 1.3)
STA A SPEEDA ;USED LATER
BRA STOD

STOD LDA A AZEYS ;DESTINATION NOT REACHED, CHECKS SIGNS
CMP A AZSIGN
BEO SAMEAZ
DIFFAZ CMP A #0CBH ;DIFF SIGNS
BEO EZ
JMP LEFT
RK JMP RIGHT
SAMEAZ CMP A #0CBH ;SAME SIGN FIND WHICH ONE PLUS
BEO YSAZ
NCAZ TST CARRY
BMI B;
JMP LEFT
B3 JMP RIGHT
YESAZ TST CARRY
BMI B4
JMP RIGHT
L4 JMP LEFT

;ELEVATION MOTOR CONTROL LOOP
STOF LDA A ELVEYS
CMP A ELSIGN
LTY ELY ;THIS CODE DUPLICATED FROM THE AZ CODE ABOVE

C-8
LDA A ELKEY
ADD A ELBCD
DAA
STA A ELMAG-1

LDA B ELMA6+i
STA A ELMAG
BRA STOY1

; ADDED TO TIGHTEN ELEVATION CONTROL LOOP TO .1 DEGREE (MOD 1.1)

STOY1
TST A BNE STOZ
CMP B #10H BHI STC'Z
LDA A #FFH STA A SFLAGE
JMP CFFLAG

STOZ
TST A BNE STOF
CMP B #00H BHI STOZ
LDA A #ETHSPD STA A SPEEDE
BRA STOH

STOF
CMP A #004H BHI STOZ
LDA A #OUASPD STA A SPEEDE
BRA STOH

STOH
LDA A ELSPD STA A SPEEDE
BRA NOEL

DIFFEL
CMP A #0CBH BEO 55
BRA 97

SAMEEL
CMP A #0CBH BEO YESEL

NOEL
TST CARRY

BMI
BRA B6

E6 BRA 56

YESEL TST CARRY

BMI
B7

B8 JMP UP
B7 JMP DOWN

; END STATE ZERO

BEGIN STATE 1

;
ST1
LDA A #QUASP D ;BEGIN STATE ONE, MANUAL DOWN
LDA B #OFFH
JSR MOTEL ;JUMPS TO SUBROUTINE COMMAND ELEVATION MOTOR

ST1A
LDA A #70H
STA A DDRA
LDA A DDRA
CMP A #77H
BNE STIB ;DOWN KEY IS NOT BEING PRESSED NOW
JSR RESTO ;RESTORE KEYBOARD BEFORE READING ANGLES (MOD 1.1)
JSR SHAENC ;DOWN KEY NOT UP YET THEREFORE READ ANGLES
BRA STIA

ST1B
CLR LFLAGE ; CLR LIMIT REACHED FLG.
LDA A #OFFH ;TURN OFF MOTOR
TAB ;CLOCKWISE MOTION STILL SET
JSR MOTEL
JSR RESTO ;RESTORE KEYBOARD PIA, AND BACK TO STATE ZERO
JMP STO

BEGIN STATE 2

BEGIN STATE 3

BEGIN STATE 4
BEGIN STATE 4

ST4
LDA A #QUASPD ;BEGIN STATE 4, LOAD A=SPEED (MOD 1.2)
LDA B #OFFH ;LOAD B WITH DIRECTION
JSR MOTAZ

ST4A
LDA A #UBOH
STA DORA
LDA A DORA
CMP A #0BH ;HAS LEFT KEY BEEN RELEASED?
BEA ST4B
JSR RESTO
JSR SHAENC
BRA ST4A

ST4B
CLR LFLAGA ;CLEAR LIMIT REACHED FLAG (MOD 1.1)
LDA A #OFFH
TAB
JSR MOTAZ
JSR RESTO
JMP ST4

;END STATE FOUR

BEGIN STATE 5

ST5
LDX #MSG4 ;DISPLAYS "ERROR INVALID ENTRY"
JSR ASCDIS
LDA B #10
JSR WAITE
JSR WAITE
JMP ST4
;BACK TO STATE 4

;END STATE 5

BEGIN STATE 6

ST6
LDX #MSG9 ;DISP "ANGLE TOO LARGE....."
JSR ASCDIS
JMP MSGB
;WAIT 1 SEC THE RETURN TO CONTROL LOOP

;END STATE 6

BEGIN STATE 7

ST7
LDX #MSG11 ;DISP "POSITIONER HALTED"
JSR ASCDIS
JMP MSGB
;WAIT 1 SEC THEN RETURN TO CONTROL LOOP

;END STATE 7

BEGIN STATE 10

ST10
STA A AZEL ;REMEMBERS WHICH KEY PRESSED (SETAZ OR SETEL)
LDX #MSG2
JSR ASCDIS
LDA #01:FL+1
STA SAVEY ;KEEP TRACK OF WHERE THINGS ARE ON THE DISPLAY
ST10A  LDA A  KFLAG  ;CLEAR KEYENTRY FLAG
BPL  ST10A
CLR  KFLAG
LDX  #MSG6
JSR  ASCDIS  ;DISPLAY "AZIMUTH"
CLR  ENTRYA
CLR  ENTRYB  ;CLEAR BOTH REGS TO BE USED WHEN PACKING ENTRIES
LDA A  KEYENT  ;GETS KEYENTRY
LDX  #SP10
JSR  ADDCAL
LDX  0,X
JMP  0,X  ;JUMPS TO NEXT STATE DETERMINED BY KEYENTRY IN A
;END STATE 10

BEGIN STATE 11

BEGIN STATE 11

ST11  STA A  AZEL  ;REMEMBERS KEY PRESSED (SETEL OR SETAZ)
LDX  #MSG1
JSR  ASCDIS  ;DISP: "ENTRY ELEVATION ANGLE"
LDX  #DISEL+1
STX  SAVEX

ST11A LDA A  KFLAG  ;WAITS FOR NEXT ENTRY
BPL  ST11A
CLR  KFLAG
LDX  #MSG7
JSR  ASCDIS  ;DISPLAYS "ELEVATION"
CLR  ENTRYA
CLR  ENTRYB
LDA A  KEYENT  ;GETS KEYENTRY
LDX  #SP11
JSR  ADDCAL
LDX  0,X
JMP  0,X
;END STATE 11

BEGIN STATE 12

BEGIN STATE 12

ST12  LDA B  #02BH  ;BEGIN STATE 12: PLUS SIGN AND MAGNITUDE
LDX  SAVEX
STA B  0,X  ;DISPLAYS PLUS SIGN
INX  ;INCREMENTS TRACKING POINTING
STA B  TEMPS  ;REMEMBERS SIGN OF ENTRY
LSH A  ;CONVERTS KEYCODE TO BCD CODE
TAB
JSR  PACK  ;ROUTINE TO PACK ENTRY
ADD B  #0J0H  ;CONVERTS BCD CODE TO ASCII
STA B  0,X  ;ECHOES KEYENTRY ON THE DISPLAY
INX
STX  SAVEX  ;REMEMBERS NEW VALUE OF TRACKING POINTER

ST12A LDA A  KFLAG  ;REMEMBERS NEW VALUE OF TRACKING POINTER
BPL  ST12A
CLR  KFLAG
LDA A  KEYENT  ;WAIT FOR ANOTHER KEYENTRY
;END STATE 12
BEGIN STATE 13

ST13 LDA B #02DH  ;DISPLAYS ENTERED MINUS SIGN
LDX SAVEX
STA B 0,X
INX  ;INC TRACKING POINTER
STX SAVEX  ;SAVE TRACKING POINTER
STA B TEMPS

ST13A LDA A #FLAG
BPL ST13A
CLR FLAG
LDA A KEYENT  ;GETS KEYENTRY
LDX #SP13  ;LOADS INDEX REGISTER WITH STATE 13 POINTER
JSR ADDCAL
LDX 0,X
JMP 0,X
;END STATE 13

BEGIN STATE 14

ST14 LSR A  ;ST14 DISPLAYS ENTERED NUM AFTER MINUS SIGN
JSR PACK
ADD B #03H
LDX SAVEX
STA B 0,X
INX  ;INC TRACKING POINTER
STX SAVEX  ;SAVE TRACKING POINTER

ST14A LDA A #FLAG  ;WAITS FOR NEXT KEYENTRY
BPL ST14A
CLR PFLAG  ;CLEAR KEYENTRY FLAG
LDA A KEYENT  ;GETS KEYENTRY
LDX #SP14  ;LOADS INDEX REGISTER WITH STATE 14 POINTER
JSR ADDCAL
LDX 0,X
JMP 0,X
;END STATE 14

BEGIN STATE 15

ST15 LSR A  ;CONVERTS KEYCODE TO BCD CODE
JSR PACK  ;ST15 DISPLAYS SECOND NUM AFTER EITHER + OR -
ADD B #03H
LDX SAVEX
STA B 0,X
INX
STX SAVEX

ST15A LDA A #FLAG
BPL ST15A  ;WAIT FOR NEXT KEYENTRY
CLR FLAG  ;CLEAR KEYENTRY FLAG
LDA A KEYENT
LDX #SP15
JSR ADDCAL
LDX 0,X
JMP 0,X
;END STATE 15

C-13
BEGIN STATE 16

ST16 
LDA B #02EH 
LDX SAVEX 
STA B 0,X 
INX 
STX SAVEX
ST16A 
LDA A #FLAG 
BPL ST16A 
CLR #FLAG 
LDA A #EVENT 
LDX #$F16 
JSR ADDCAL 
LDX 0,X 
JMP 0,X 

END STATE 16

BEGIN STATE 17

ST17 
LSR A 
JSR PACK
ADD B #000H 
LDX SAVEX 
STA B 0,X 
INX 
LDA A #27H 
STA A 0,X 
INX 
STX SAVEX 
LCX SAVEX 
SIA B 
INX 
LDA A #27H 
STA A 0.X 
JSR ADDCAL 
LDX 0,X 
JMP 0.X 

END STATE 17
BEGIN STATE 19

BEGIN STATE 19

ST18
CLR MFLAG ; CLEARS MOTOR FLAG: ST18 GO TO CONTOL LOOP
CLR LFLAGA ; CLEARS LIMIT REACHED FLAG
CLR LFLAGA
JMP ST0

END STATE 18

BEGIN STATE 19

ST19
LDA A #OFFH ; ST19 DISABLES CONTROL LOOP (MOD 1.1)
STA A MFLAG ; SETS MOTOR FLAG SO CNTRL LOOP DISABLED
JSR ALSTOP
TST LFLAGA ; CHECK FOR AZ LIMIT REACHED
BEO ST19A ; GO IF NOT REACHED
CLR A ; AZ LIMIT REACHED
LDA B DFLAGA ; GET CURR DIRECTION STATUS
COM B ; GET OFF DIRECTION
JSR MOTAZ
LDA B #10
JSR WAITE ; ALLOW TIME FOR AZ MOTOR TO REPOSITION GIMBAL
LDA A #OFFH
LDA B DFLAGA
JSR MOTAZ ; STOP MOTOR, LIMIT NO LONGER EXCEEDED
JMP ST0

ST19A
CLR LFLAGA
TST LFLAGA ; CHECK FOR EL LIMIT REACHED
BEO ST19B ; GO IF EL LIMIT NOT EXCEEDED
CLR A ; EL LIMIT REACHED
LDA B DFLAGA ; GET CURR DIRECTION
COM B ; GET OFF DIRECTION
JSR MOTEL
LDA B #10
JSR WAITE ; WAIT FOR EL MOTOR TO FINISH MOVE
LDA A #OFFH
LDA B DFLAGA
JSR MOTEL ; STOP EL MOTOR, LIMIT NO LONGER EXCEEDED
JMP ST0

ST19B
JMP ST7 ; DISPLAYS "POSITIONER HALTED", WAIT 1 SEC, STO

END STATE 19

BEGIN STATE 20

BEGIN STATE 20

ST20
LDX #MSG9
JSR ASCDIS ; DISPLAY "ENTER PROG NUMBER"
LDX #DISEL+5
STX SAVEX

ST20A
LDA A PFLAG
BPL ST20A ; WAIT FOR KEVENT
CLR PFLAG
CLR ENTRAY
CLR ENTRAYB
LDX #MSG10
JSR ASCDIS ; DISPLAY "PROG?
LDA A KEVENT
LDX #M20
JSR ADDCAL

C-15
BEGIN STATE 20

BEGIN STATE 21

BEGIN STATE 22

BEGIN STATE 23

C-16
LDX #SP23
JSR ADCAL
LDX 0,X
JMP 0,X
:END STATE 23

BEGIN STATE 24

BEGIN STATE 24

ST24

LDA B #02EH
LDX SAVEX
STA B 0,X
INX
STX SAVEX

ST24A

LDA A KFLAG
BPL ST24A
LDA A KEYENT
LDX #SP24
JSR ADCAL
LDX 0,X
JMP 0,X
:END STATE 24

BEGIN STATE 25

ST25

LSR A ;ST25-DIS LAST ENTERED NUM OF 3 FOR RASTER SCAN
TAB
JSR PACK
ADD B #020H
LDX SAVEX
STA B 0,X
INX
LDA B #027H
STA B 0,X
LDX #DIS+5 ;ECHOES ENTER NUM AND DEGREE MARK
STX SAVEX ;STORES TRACKING POINTER
LDA A PROGL
CMP A #0042H
BED ST25B ;BRANCH IF PROGL IS "B"
CMP A #047H
BED ST25C ;BRANCH IF PROGL IS "C"
LDX ENTRYA
STX PROGA
LDA B #10
JSR WAITE ;1 SEC WAIT

;ADDITION TO ST25. CHECKS TO SEE IF PROG #4 IS
;BEING IMPLEMENTED (MOD 1.1)

LDA B PROGN ;GET CURR PROGRAM COUNTER
CMP A #34H ;COMPARES WITH ASCII "4"
BED ST25A ;GO IF STATE=PROGRAM 4
LDA #MSG3
JSR ASCDIS ;CLEARS DISPLAY
LDX #MSG30
JSR ASCDIS ;DISPLAY "PROG :ENTER"
LDA A PROGN ;DISPLAYS THE PROGRAM NUM
LDA A #04CH

C-17
STA A PROGL ;STORES PROGRAM LETTER
JMP ST21A
LDX ENTRYA
STX PROGB

;ADDITION TO ST250. CKS SEE IF PROG3 BEING IMPL'D

;MOD (1.1)

LDA B PROGN
CMP B #33H ;ASCII "3"
BEQ ST25A ;GO IF STATE=PROG 3

LDA B #10
JSR WAITE
LDX #MSG2
JSR ASCDIS
LDX #MSG10
JSR ASCDIS
LDA A PROGN
STA A DISA2+5
LDA A #043H ;DISPLAY "C"
STA A DISB+3
STA A PROGL ;STORES CURR PROG LETTER
JMP ST21A ;JUMPS TO ENTER "C"

LDA A #10
JSR WAITE
LDX ENTRYA
STX PROGC

;MOD (1.2) PATTERN NUM ONE

LDA A PROGN
CMP A #31H ;BR IF IN PROG #1
BEQ ST001
CMP A #32H
BEQ ST002 ;BR IF IN PROG #2

LDA A #334
BLO ST003 ;BR IF IN PROG #3
JMP ST29 ;GO TO PROG #4

JMP ST26 ;GO TO PROG #1
JMP ST27 ;GO TO PROG #2
JMP ST29 ;GO TO PROG #3

;END STATE 00

END STATE 25

LDA A PROGN
CMP A #11H ;GET PROG NUM
BEC ST001 ;BR IF IN PROG #1
CMP A #12H
BEC ST002 ;BR IF IN PROG #2

LDA A #334
BLO ST003 ;BR IF IN PROG #3
JMP ST29 ;GO TO PROG #4

JMP ST26 ;GO TO PROG #1
JMP ST27 ;GO TO PROG #2
JMP ST29 ;GO TO PROG #3

;END STATE 00

BEGIN STATE 26

;CLEAR X REG AND PROG FLAG

C-18
LDX PROGC
STX AZKEY
LDX PROGC
STX ELKEY
LDX #ST26A
STX STADDR
CLR PROCNT
DEC PFLAG
JMP STO

ST26A LDA B #02BH
STA AZKEYS
LDX #ZD2BH
STA AZKEYS
LDX #ST26A
SIX STADDR
CLR PROCNT
DEC PFLAG
JMP STO

ST26B LDA B ELKEY
LDX #PROGC
STA #X REG POINTS TO DESTINATION
JSR BCDSUB
STA #CURR EL POS)-(DEST)
CMP A #1
BHI WEIRD
STA #CURR EL INCREMENTAL ANGLE
DAA
STA AZKEYS

ST26B0 STA B ELKEY
JMP #ST26B

ST26B1 STA B ELKEY+1
STA B ELKEY+I
TST CARRY
BEO SF2601

ST26C LDA B #ZDH
STA AZKEYS
LDX #ST26A
SIX STADDR
CLR PROCNT
STA AZKEYS
LDX PROGC
STA STADDR
SIX STADDR
CLR PROCNT
JMP STO

C-19
INCR PROGCTN  ;INC PROGRAM COUNTER
JMP ST0
ST26D CLR FFLAG  ;CLEAR PROGRAM FLAG
JMP ST7  ;DISPLAY "POSITIONER HALTED"
            ;GO TO CONTROL LOOP AND DO NOT COME BACK
END STATE 26

BEGIN STATE 27

ST27-PATTERN #2

ST27
LDX #0  ;THIS CODE SAME AS ABOVE
STX MFLAG
LDX PROGA
STX AZKEY
LDX PROGC
STX ELKEY
LDX #002BH
STX AZKEYS
LDX #ST27A
STX STAUR
CLR PROCNT
DEC FFLAG
JMP ST0

ST27A
LDA B #0CMH
STA B ELKEYS
LDA B AZSIGN
CMP R #20H
BEO ST27A1
LDA A AZCD
LDA B AZCD+1
LDX #PROGC
JSR BCDSUB
CMP A #1
BHI WEIRD1
BRA DONE1

WEIRD1
LST CARRY
BMI ST27A1
DONE1
LDX #ST27B
BRA ST27A2
ST27A1
LDX #ST27B
ST27A2
STX STAUR
INC PROCNT
JMP ST0

ST27B
LDA B AZSIGN
CMP B #2CMH
BEO ST27B0
LDA A AZKEY+1
ADD A PROGB+1
DAA
TAB
LDA A AZKEY
ADC A PROGB
DAA
BRA ST27B01

ST27B0
LDA A AZKEY
LDA B AZKEY+1
LDX #PROGB  ;GET ENTERED INC COUNTER
JSR BCDSUB
ST27B01
STA A AZKEY  ;ENTER IN THIRD POINT
STA B AZKEY+1
TST CARRY
BNE ST27H

C-20
LDA B #2BH
STA B AZKEYS
ST2781 LDA B PROCNT
CMP B #3
BEC ST2782
LDX #ST27C
INC PROCNT
BRA ST27C
ST2782 CLR PROCNT
LDX #ST27A
ST2783 STX STADDR
JMP ST0
ST27C LDA B #2BH
STA B ELKEYS
LDX #ST27B
STX STADDR
INC PROCNT
JMP ST0
ST27D CLR FFLAG
JMP ST7
;END STATE 27
BEGIN STATE 29

;ST29-GENERATES ONE CIRCLE W/RADIUS "A" (MOD 1.2)

ORG 2C00H

ST29
LDX #0
STX PROANG ;INIT ANGLE COUNTER
STX MFLAG ;CLEAR PROGFLAG AND MOTFLAG
LDA A PROGA ;GET ENTERED ANGLE "A"
LDA B PROGB+1
JSR DIVID100 ;DIVIDE ANGLE BY 100 (XX,XX=00XX.)
JSR BCDBIN ;CONVERT BCD TO BIN
STA A RADIUS ;SAVE TWO BYTE RESULT
STA B RADIUS+1
DEC PFLAG

ST29A
LDX PROANG ;GET UPDATED ANGLE COUNTER
CPX #361 ;COMPARE PROANG TO 361 DEGREES
BEC ST29B ;SR IF PATTERN FINISHED
LDX #PROANG ;SET X POINTER TO ANGLE POINTER
JSR TRGVALUE ;GET SIN,COS
LDX #ST29A
STX STADDK
LDX PROANG ;BEGIN UPDATING PROANG
INX STX ;INC PROANG BY 1 DEG
STX PROANG
JMP ST7

ST29B
CLR PFLAG
JMP ST7

END STATE 29

BEGIN STATE 30

;ST30-SET NEGATIVE ELEVATION LIMIT

LDX #MSG16
JSR ASCDIS ;DISPLAY "NEG EL LIMIT"
LDX #DISEL+6
STX SAVEX ;SAVE DISPLAY TRACKING POINTER
LDA A NELIM ;GET CURRENT NEGATIVE ELEVATION LIMIT
LDA B NELIM+1
JSR UCDDIS ;DISPLAY CURRENT NEG EL LIM

C-22
LDA A       KEYENT
LDX #SP32
JSR ADDCAL
LDX 0,X
JMP 0,X
ST32B
LDX ENTRPY
STX NAZLIM  ;UPDATE NEGATIVE AZIMUTH LIMIT
BRA ST32A
;END STATE 32

;BEGIN STATE 33
;ST33-SET POSITIVE AZIMUTH LIMIT (MOD 1.1)
ST33
LDX #MSG19
JSR ASCDIS  ;DISPLAY "POS AZ LIMIT"
LDX #DISEL+6
STX SAVEX
LDA A PAZLIM
LDA B PAZLIM+1
JSR BCDDIS ;DISPLAY CURRENT POS AZ LIMIT
JSR MOVED
LDX #ST33B
STX STADOR
ST33A LDA A KFLAG
BPL ST33A
CLR KFLAG
LDA A KEYENT
LDX #SP33
LDX 0,X
JMP 0,X
ST33B
LDX ENTRPY
STX PAZLIM
BRA ST33A
;END STATE 33

;BEGIN STATE 34
;ST34-INPUT NUMBERS FOR SETTING LIMITS--1ST
ST34
LSR A       ;BEGIN INPUT, CONVERT KEYCODE TO BCD CODE
TAB
JSR PACK   ;PACKS BCD INPUT INTO PACKED BCD FORM
ADD B #0XH
LDX SAVEX
STA B 0,X
INX
STX SAVEX
INC TRACKING POINTER
ST34A
LDA A KFLAG
BPL ST34A
CLR KFLAG
LDA A KEYENT
LDX #SP34
JSR ADDCAL
LDX 0,X
JMP 0,X
;END STATE 34

;BEGIN STATE 35
;ST35-SET LIMITS-2ND
; ST35-INPUT BCD NUMBER—2ND

ST35
  LSR A  ; CONVERT KEYCODE TO BCD CODE
  TAB
  JSR  PACK
  ADD B  #20H
  LDX  SAVE
  STA B  0, X
  INX
  STX  SAVE

ST35A
  LDA A  FLAG
  BPL  ST35A
  CLR  FLAG
  LDA A  KEYENT
  LDY  #SFC5
  JSR  ADDCAL
  LDX  0, X
  JMP  0, X

; END STATE 35

; BEGIN STATE 36

; ST36-INPUT DECIMAL POINT

ST36
  LDA B  #2EH  ; CONVERT INPUT TO BCD CODE
  LDY  SAVE
  STA B  0, X
  INX
  STX  SAVE

ST36A
  LDA A  FLAG
  BPL  ST36A
  CLR  FLAG
  LDA A  KEYENT
  LDY  #SFC5
  JSR  ADDCAL
  LDX  0, X
  JMP  0, X

; END STATE 36

; BEGIN STATE 37

; ST37-INPUT LAST BCD CHARACTER—3RD

ST37
  LSR A  ; CONVERT INPUT KEYCODE TO BCD CODE
  TAB
  JSR  PACK
  ADD B  #20H
  LDX  SAVE
  STA B  0, X

ST37A
  LDA A  ENTRYA
  LDA B  ENTRYB
  JSR  TSTANG  ; TEST FOR ANGLE LIMIT CONDITION
  LDY  STADOR
  JMP  0, X

; END STATE 37

; BEGIN STATE 38

; ST38-SET AZ GIMBAL SPEED

ST38
  LDY  #MSG21
  JSR  ASCDIS  ; DISPLAY "AZ GIMBAL SPEED"
  LDX  ""
A

; GET LAST USER SELECTED GIMBAL SPEED
LDA A
STA A 0X
LDX #ST38B
STX STADDR
; SAVES RETURN ADDRESS
ST38A
LDA A
BPL ST38A
CLR KFLAG
LDA A KEYENT
JSR ADDCAL
LDX 0X
JMP 0X
ST38B
LDA B
STA A 0X
STA A
LDX #ST39B
STX STADDR
LDA A
BPL ST39A
CLR KFLAG
STA B
JSR ADDCAL
LDX 0X
JMP 0X
ST39B
LDA B
STA A 0X
STA A
LDX #ST40B
STX STADDR
LDX #ST40A
JMP 0X
C-26
SUBROUTINE CONVERSION OF 32 BIT FLOATING POINT NUMBER INTO A 32 BIT PACKED BCD ROUTINE. ROUTINE ASSUMES 32 FLT PT NUM="FPT32" ENTRY: X-REG POINTS TO LOCATION OF RESULT EXIT: BCD RESULT SPECIFIED BY X REG, REG A,B DESTROYED

FPTBCD STX SAVEX1 ;SAVE RESULT POINTER
CLR INTREG ;INITIALIZES INTEGER Registers
CLR INTREG+1
CLR 0,X ;INIT BCD SIGN VAL
LDX #FPT32 ;SET POINTER TO 32 BIT NUM
LDA A,0,X ;GET EXP PART OF 32 BIT NUM
BEO FPT7 ;DONT SHIFT IF EXP=0
BPL FPT1 ;BR IF 32 BIT NUM >0
LDX SAVEX1 ;IT IS NEG, FIX BCD SIGN BYTE TO BE NEG.
DEC 0,X ;SIGN BYTE NEGATIVE
LDX #FPT32 ;RESET DATA POINTER
FPT1 ASL A ;CONVERTS 7 BIT 2S COMP TO 8 BIT COMP
ASR A
BPL FPT4 ;BR TO POS SHIFT ROUT IF 32 BIT # POS
FPT2 LSR 1,X ;MOVES BINARY POINT OVER TO LEFT
ROR 2,X ;(MOD 1.4)
ROR 3,X ;(MOD 1.4)
DEC B ;DEC BIT COUNTER
BNE FPT3 ;BR IF UNDERFLOW DOES NOT OCCUR
BRA FPT7 ;BR TO END OF ROUTINE
FPT3 INC A ;ADJUSTS THE EXP
BNE FPT2 ;BR THRU LOOP UNLESS BIN POINT ADJUSTED
BRA FPT7 ;BR TO CONVERSION PART OF ROUTINE
FPT4 LDA B #17 ;INIT BIT COUNTER (MOD 1.4)
FPT5 ASL 2,X ;BEGIN POS SHIFT ROUTINE
ROL 2,X ;ROTATE THIRD BYTE OF 32 BIT NUM
ROL 1,X ;ROTATE SECOND BYTE OF 32 BIT NUM
ROL INTREG+1 ;ROTATES 32 BIT NUM INTO INTEGER REG
ROL INTREG
DEC B ;DEC BIT COUNTER
BNE FPT6 ;BR IF OVERFLOW HAS NOT OCCURED
SEV ;SETS OVERFLOW BIT TO DENOTE AN ERROR
BRA FPT8 ;BR TO END OF ROUTINE
FPT6 DEC A ;ADJUST EXP
BNE FPT5 ;BR THRU LOOP UNLESS BIN PT ADJUSTED
FPT7 LDA A 1,X ;GET BIN FRACTION IN ACCA
LDA B #2,X ;(MOD 1.4)
PSH A ;SAVE A
PSH B ;SAVE B (MOD 1.4)
LDX SAVEX1 ;SETS DATA POINTER
INX LDA INTREG ;RETRIEVES THE BIN INTEGER
INX LDA B INTREG+1
PSH B INBCDED ;CONVERT INTEGER PART
PSH A
INX
INX
INX
PUL B ;(MOD 1.4)
PUL A
Bsr BINFPT ;CONVERT FRACTIONAL PART
FPT8 RTS ;RETURN

"BINBCDED"-SUBROUTINE
BINARY TO PACKED BCD CONVERSION ROUTINE
LOAD ACCA,ACCB WITH A 16 BIT BIN NUM
LOAD INDEX REG WITH ADDRESS OF MS BYTE OF WHERE PUT PACKED BCD ANS

C-27
RETURNS WITH PACKED INFO IN SPEC LOCATION AND IN ACCA, ACCB
THE LEAST 4 MS DECIMAL VALS WILL BE CONTAINED IN THE PACKED BCD ANS;

BINBCDED STX SAVEX ;SAVE DATA POINTER
LDX #K10K ;INITS X-REG FOR 1ST BCD Conv Const
CLR ENTRY1
CLR ENTRY2
ZVDEC1 CLR SAVEX ;CLEAR BCD CONVERSION COUNTER
ZVDEC2 SUB B 1,X
SBC A 0,X
BCS ZVDEC5 ;BR IF SUB PRODUCES OVERFLOW
INC SAVEX ;DEC CHAR BEING BUILT, INC SAVEX
BRA ZVDEC2
ZVDEC5 ADD B 1,X
ADC A 0,X ;RESTORES PARTIAL RESULT UPON OVERFLOW
PSH A ;SAVE ACCA
LDA A SAVEX ;GETS BCD CONVERSION COUNTER
BSR PACKED ;PACKS NEWLY FORMED BCD CHARACTER
PUL A ;RESTORES ACCA TO FORMER VALUE
INX
INX
CPX #K10K+10 ;TESTS TO SEE IF LAST CONSTANT HAS BEEN USED
BNE ZVDEC1
LDA A ENTRY1 ;LAST CHARACTER HAS BEEN REACHED
LDA B ENTRY2
LDX SAVEX
STA A 0,X
STA B 1,X ;SAVE 16
RTS ;END BINBCDED SUBROUTINE

"PACKED" SUBROUTINE
PACKS BINARY NUMBER INTO BCD FORM
ACCA SHOULD CONTAIN THE UNPACKED BCD FORM
ROUTINE DESTROYS CONTENTS OF ACCA

PACKED ASL ENTRY2 ;ONE BIT LEFT SHIFT WITH ZERO FIL
ROL ENTRY1
ASL ENTRY2
ROL ENTRY1
ASL ENTRY2
ROL ENTRY1
ROL ENTRY1 ;SHIFTS 16 BIT BINARY INFO OVER ONE CHAR
ADD A ENTRY2
STA A ENTRY2 ;ENTRY2 FORM="X0", PACKS ANOTHER UNPACKED FORM
RTS ;RETURN FORM SUBROUTINE

"BINFPT" SUBROUTINE
CONVERSION OF FRACTIONAL PART OF BINARY NUM TO PACKED BCD
LOAD FRACTIONAL PART IN ACCA BEFORE EXECUTING
ACCB IS USED IN CALCULATION
ROUTINE EXITS WITH BCD ANSWER (4 DEC PLACES) IN ACCA, ACCB

BINFPT STX SAVEX ;SAVE DATA POINTER
STA A SAVEX ;SAVE FRACT PART
STA B SAVEB ;(MOD 1.4)
LDA B #16 ;(MOD 1.4)
STA B SAVEC ;SAVE BIT COUNTER (MOD 1.4)
LDX #K10K ;SET POINTER IN ACCX AT FIRST BYTE OF CONSTANTS
CLR A

C-28
CLR B
STA A TEMPA ;SAVE ACCA TEMPORARILY
ABD SAVED ;LOOK AT NEXT BIT
ROL SAVEA
BCC BIN2 ;BR PAST LOOP IF C=0
TBA ;OUT ACCB INTO ACCA
ADD A 1,X ;C=1,ADD IN CONSTANT
DAA
TAB LDA A TEMPA ;RETRIEVE ACCA FROM TEMP STORAGE
ADD A 0,X
DAA
BIN2 INX ;INC ACCX TO NEXT CONSTANT
INX
DEC SAVEC ;DEC BIT COUNTER
BNE BIN1 ;BR THRU LOOP UNTIL B BITS ARE SHIFTED
LDX SAVEX ;RETRIEVE DATA POINTER
STA A 0,X ;SAVE 16 BIT PACKED BCD CHARACTE
STA B 1,X
RTS ;RET FROM SUBROUTINE

;"DIVID100" SUBROUTINE
; DIVIDES BCD VAL BY 100
; ENTRY: ACCA,ACCB CONTAIN 16 BIT BCD NUMBER
; EXIT: ACCA,ACCB CONTAIN 16 BIT BCD RESULT

"DIVID100" TAB ;THROWS AWAY THE FRACTIONAL PART
CLR A ;ACCA,ACCB="00XX"
RTS ;RETURN FROM SUBROUTINE

;"CHSIGN" SUBROUTINE
; CHANGES THE SIGN OF A TWO BYTE COORD FROM ONE STATE
; TO THE OTHER
; ENTRY: ACCB CONTAINS SIGN INFO,X REG POINTS TO LOCATION OF
; SIGN INFO
; EXIT: ORIGINAL SIGN INFO AUTOMATICALLY CHANGED,ACCB DESTROYED

"CHSIGN" CMP B #2BH ;FIND OUT SIGN
BEQ CHSIGN1 ;BR IF PLUS
LDA B #2BH ;SIGN IS PLUS ASCII IS 2BH
BRA CHSIGN2

CHSIGN1 LDA B #2DH ;SIGN IS MINUS ASCII IS 2DH
CHSIGN2 STA B O,X ;UPDATE SIGN VAL
RTS ;RETURN

;"CONSIGN" SUBROUTINE
; GETS SIGN OF 32 BIT FP # AND FINDS ASCII EQUIV
; ENTRY: ACCB CONTAINS SIGN PORTION OF FP #
; X REG CONTAINS LOCATION OF RESULTING SIGN
; EXIT: APPROPRIATE SIGN IS LOCATED, ACCE DESTROYED

"CONSIGN" BMI CONSIGN1 ;BR IF NEG
LDA B #2BH ;SIGN IS PLUS ASCII = 2BH
BRA CONSIGN2

CONSIGN1 LDA B #2DH ;SIGN IS MINUS ASCII IS 2DH
CONSIGN2 STA B 0,X ;UPDATE SIGN VAL
RTS ;RETURN

;"TRGVALUE" SUBROUTINE - SIN,COS OF BINARY ANGLE
; ENTRY: ACCA=# BYTES, X-REG POINTS TO ADDR OF BIN ANGLE

C-29
TRGVALUE

```
LDX #RADIANS ;X-REG POINTS TO PI/180 CONST
LDA B #2H ;ACCB=#BYTE TO BE PUT AT TOS
BSR PUSH ;PUSH ANGLE ONTO APU STACK
BSR FLT6 ;CONVERT ANG TO 32 BIT FLT PT FORM
LDX #RADIUS ;X-REG POINT TO RADIUS 16 BIT FIXED POINT
LDA B #4H ;
BSR PUSH ;
BSR FMUL ;BIN ANG * PI/180 = ANGLE IN RADIANS RESULT TOS
BSR PTOF ;DUP BIN ANGLE AT NOS
BSR COS ;TAKE COS OF ANGLE
LDX #RADIUS ;X-REG POINT TO RADIUS 16 BIT FIXED POINT
LDA B #2H ;
BSR PUSH ;
BSR FLT6 ;CONVERT RADIUS TO 32 BIT FL PT #
BSR PTOF ;PUSH 32 BIT TOS TO NOS
BSR POPF ;32 BIT APU STACK POP
BSR FMUL ;RADIUS*COS(ANG)=ELEVATION
LDX #FPT32
LDA B #4H ;
BSR PULL ;FULL ELEVATION FROM APU STACK
LDX #ELRESULT ;X-REG POINTS TO LOCATION OF RESULT
JSR FPTBCD ;CONVERT 32 BIT FP TO 4 BYTE BCD RESULT
BSR SIN ;RADIUS*SIN(ANG)=AZ
BSR FMUL ;RADIUS*SIN(ANG)=AZ
LDX #FPT32
LDA B #4H ;
BSR PULL ;
LDX #AZRESULT ;X-REG POINTS TO RESULT
JSR FPTBCD ;CONVERT 32 BIT FP TO 4 BYTE BCD RESULT
LDX #ELKEYS
LDA B #2H ;GET MSBYTE OF MAG VAL OF FPT
LDA B #4H ;GET LSBYTE OF MAG VAL OF FPT
STA A ELKEY ;UPDATE EL COORD
STA B ELKEY+1 ;
LDX #ELRESULT
LDA A 2,X ;GET MSBYTE OF MAG VAL OF FPT
LDA A 3,X ;GET LSBYTE OF MAG VAL OF FPT
STA A ELKEY ;UPDATE EL COORD
STA B ELKEY+1 ;
RTS ;RETURN
```

ENTRY: ACCB CONTAINS #BYTES TO PUSH ONTO STACK
EXIT: ACCX CONTAINS ADDRESS OF MSBYTE OF DATA TO PUSH
DATA WILL BE PLACED ON APU STACK SUCH THAT MSB OF WILL BE TOS. X-REG DESTROYED, ACCB DESTROYED

```
PUSH B ;SAVES THE NUMBER OF BYTES
PUSH1 INX ;BR THRU TILL LAST BYTE REACHED
DEC B ;
BNE PUSH1
STX TEMPX
PUSH2 DEX ;RETRIEVES #BYTES TO BE PUSHED
LDA A 0,X ;ACCESS NEXT ADD TO PUT ONTO APU STACK
STA A APUDATA ;ENTER CURR BYTE OF DATA ONTO APU STACK
```
ULL B
BNE PUSH2
LDX TEMPX
RTS
;END PUSH

"FLTS" SUBROUTINE
CONVERT 16 BIT FIX PT TO 32 BIT FP
ENTRY: 16 FIXED # ON TOS
EXIT: WHEN APU FINISHED

FLTS LDA B #1DH ;LOADS IN FLOAT OPCODE
BRA TSTEND ;WAIT TILL APU FINISHED

"PTOF" SUBROUTINE
ENTRY: 32 BIT FP # AT TOS
EXIT: WHEN APU DONE

PTOF LDA B #17H
BRA TSTEND

"COS" SUBROUTINE
ENTRY: 32 BIT FLOATING COSINE
EXIT: WHEN APU DONE

COS LDA B #3H ;LOAD IN COS COMMAND
BRA TSTEND

"POPF" SUBROUTINE - POP NOS INTO TOS
ENTRY: 32 BIT FP # AT TOS
EXIT: WHEN APU DONE

POPF LDA B #1BH
BRA TSTEND

"FMUL" SUBROUTINE - 32 BIT FLOATING POINT MULTIPLIER
ENTRY: 2 #S ON TOS AND NOS
EXIT: WHEN APU DONE

FMUL LDA B #1CH
BRA TSTEND

"PULL" SUB - REMOVES B BYTES OF DATA FROM THE STACK
ENTRY: ACCB = # BYTES TO BE PULLED
ACCB = ADDRESS TO WHERE TOS TO BE PLACED
EXIT: WHEN DONE

PULL LDA A APUDATA
STA A 0,X
INX
DEC B
BNE PULL
RTS

"SIN" SUB - 32 BIT FP SIN
ENTRY: 32 BIT FP AT TOS
EXIT: WHEN DONE

SIN LDA B #2H
BRA TSTEND

"XCHF" SUB - EXCH 32 BIT OPERANDS TOS AND NOS
ENTRY: BOTH #’S MUST BE ON APU STACK BEFORE XEW
EXIT: WHEN DONE

XCHF  LDA B  #19H
      BRA  TSTEND

"FADD" SUB - 32 BIT FLOATING POINT ADDITION
ENTRY: BOTH #’S ON STACK
EXIT: BOTH #’S ON STACK
FADD  LDA B  #10H
      BRA  TSTEND

"FSUB" SUBROUTINE - 32 BIT FP SUBTRACT
ENTRY: BOTH #’S ON TOS AND NOS
EXIT: RESULT ON TOS
FSUB  LDA B  #11H
      BRA  TSTEND

"FDIV" SUB - DIVIDE 2 32 BIT FP #’S
ENTRY: 2 32-BIT #’S ON NOS AND TOS
EXIT: RESULT ON TOS
FDIV  LDA B  #13H
      BRA  TSTEND

"TSTEND" SUBROUTINE - LOOPS UNTIL ENDFLAG FROM APU LOW (PA7 OF PIA2)
ENTRY: ISSUE COMMAND TO APU
EXIT: WHEN APU DONE
TSTEND  STA A  APUSTAT  ; ISSUES COMMAND TO APU
TSTENDI LDA A  DDRA3  ; ISSUE ENDFLAG
      BPL  TSTEND1 ; LOOPS TILL LOW
      LDA A  APUSTAT  ; CLEAR FLAG
      RTS

"RESTO" SUBROUTINE - RETURNS THE KEYBOARD TO AN INITIALIZED STATE
SO THAT ANY KEY PRESSED WILL GENERATE AN INTERRUPT
RESTO  LDA A  #0FH
      STA A  DDRA  ; RESTORE THE ROWS OF KBD FOR NEXT KEY PUSHED
      LDA A  DDRA  ; CLEAR IRQ BITS IN CRA
      LDA A  #0FFH
      STA A  MFLAG  ; DIS CONTROL LOOP
      RTS

"MOVED" SUBROUTINE - INSERTS A DECIMAL POINT IN FRONT OF LAST
BCD CHARACTER (USED IN ST30-33)
MOVED  LDA A  DISEL+8
       LDA B  #2EH
       STA B  DISEL+8
       STA A  DISEL+9
       RTS

"CPFLAG" SUBROUTINE - (MOD 1.1)
CPFLAG  LDA A  JFLAGA
       CMP A  #0FFH
       BNE  CPI
       TST  PFLAG
       BNE  CPI
       JMP  0,X
       JMP  STO  ; Jumps Back to the Programmed Control Loop
       CPI  JMP  STO  ; PROGRAM FLAG CLEARED, JOTO CNTL LP

"TSTANG" SUBROUTINE
COMPARES THE TWO ACC"S WITH THE CONTENTS OF THE INDEX REGISTER. RETURNS FROM SUB IF CONTENTS OF INDEX REGISTER ARE > THE CONTENTS OF THE ACC"S BRANCHES TO ERROR 6 IF NOT

TSTANG LDX #LIMI
JSR BCDSUB
TST CARRY
BMI NOPE
JMP ST6 ;BR IF ENTERED ANGLE EXCEEDS LIMIT OF +/- 40.0
NOPE RTS

"BCDBIN" SUBROUTINE
CONVERTS FOUR BINARY CODED DECIMAL DIGIT TO A BINARY EQUIVALENT. THE BCD DIGITS ARE PACKED TWO PER BYTE. THE BINARY RESULT OCCUPIES TWO BYTES THE BCD DIGITS ARE LOADED INTO THE ACCA AND ACCB (MSD-ACCA) AND THE BCDBIN SUBROUTINE IS CALLED, THE ROUTINE EXITS WITH THE BINARY RESULT IN ACCA AND ACCB (MOD 1.1)

BCDBIN STA A SAVEI ;SAVE 2 BCD VALS
CLR BINUPR
TBA AND B #OFH ;SAVE ONLY LS BCD VAL
LSR A
LSR A
LSR A
LSR A

TENLP BED DOHUND ;GO DOHUND WHEN TEN IS ZERO
ADD B #10 ;ADD 10 TO BINARY TOTAL
DEC A ;DEC TENS DIGIT AND
BRA TENLP ;REPEAT UNTIL 0

DOHUND CLC LDA A SAVEI ;GET HUN IN THOU DIGIT
AND A #OFH ;SAVE ONLY HUN DIGIT
BEO DOHUND ;DO HUN IF HUN IS 0
ADD B #100 ;ADD 100 TO BINARY VAL
BCC HUNDO
INC BINUPR ;ADD 256 TO BINARY UPPER VAL
HUNDO DEC A ;DEC HUN DIGIT ONE
BRA HUNLP ;REPEAT TIL 0

HUNLP BEO HUNDO ;DO THOU IF HUN IS 0
ADD B #100 ;ADD 100 TO BINARY VAL
BCC HUNDO
INC BINUPR ;ADD 256 TO BINARY UPPER VAL
HUNDO DEC A ;DEC HUN DIGIT ONE
BRA HUNLP ;REPEAT TIL 0

THOUO LDA A BINUPR ;GET BIN UPPER VALUE
BRA XITBIN

HUNO0 LDA A BINUPR ;GET HUN DIGIT
LSR A
LSR A
LSR A
LSR A

DOHUND LDA A SAVEI ;SAVE HUN DIGIT
BNE THOUO ;BR IF THOU DIGIT = 0
LDA A BINUPR ;GET BIN UPPER VALUE
BRA XITBIN

THOUO LDA A BINUPR ;GET BIN UPPER VALUE
CLC ;RESET CARRY
ADD B #232 ;ADD 232 TO BINARY LOWER
ADC A #2H ;ADD 32 TO BINARY UPPER
DEC A ;DEC THOU DIGIT
BNE THOULP ;REPEAT TILL THOU DIGIT = 0

THOULP LDA A BINUPR ;GET BIN UPPER VALUE
CLC ;RESET CARRY
ADD B #232 ;ADD 232 TO BINARY LOWER
ADC A #2H ;ADD 32 TO BINARY UPPER
DEC A ;DEC THOU DIGIT
BNE THOULP ;REPEAT TILL THOU DIGIT = 0

XITBIN RTS

"PACK" SUBROUTINE
PACKS BINARY #S INTO BCD FORM
ACCA=UNPACKED BCD FORM
DESTROYS ACCA

PACK STA A ENTRYB ;ENTRYB LOOKS LIKE "XO"

C-33
"ALSTOP" SUBROUTINE
ROUTINE THAT STOPS BOTH MOTORS FOR EXCEEDING ANGLE LIMIT

ALSTOP
LDA A #OFFH
STA A LSBSAZ ;STOPS AZ MOTOR
STA A LSBSEL ;STOPS EL MOTOR
RTS

"BCDSUB" SUBROUTINE
SUBTRACT 2 16-BIT BCD PACKED #"S
SUBTRACTS INDEXED ADDRESS FROM ACCA,ACCB
INDEX REG CONTAINS STARTING ADDRESS
ACCA,ACCB CONTAINS # TO BE SUBTRACTED FROM
RETURN RESULT IN ACCA,ACCB

***************
# 9999 #
# ------ #
#  DIFF  #
#  +1   #
# ------ #
#  DIFF+1 #
# + BCD# #
# ------ #
# ANSWER #
***************

BCD TSTS OV CONDITION
RCC TSTS NO OV CONDITION

BCDSUB
CLR CARRY ;RESET CARRY VALUE
CMP A 0,X ;IS CONTENTS OF ACCA BIGGER ?
BHI SUBT ;BR IF MINUEND>SUBTRAHEND
BNE SWAP ;BR IF MIN>MINUEND
CMP B 1,X ;MSBYTE OF MIN>MSBYTE OF SUBTRAHEND
BHI SUBT ;BR IF MSBYTE OF MIN>SUBTRAHEND
BFI SUBT ;BR IF MSBYTE OF MIN>MSUBTRAHEND OK TO SUBTRACT AS IS

SWAP
PSH B ;SAVE MIN TEMPORARILY
PSH A
DEC
LDA A 0,X ;GET SUBTRAHEND
LDA B 1,X
TSA

SUBT
STA A TEMPA ;SUBTRACT SMALLER FR GREATER
STA B TEMPB
LDA A #OFFH
TAB
SUB A 1,X ;RESULT => ACCA,ACCB "9999"
SUB B 0,X ;SUBTRACT SMALLER NUM FROM 9999
SEC
ACC A TEMPB ;LS BYTE OF DIFF+LSBYTE OF GREATER NUM +1
DAA
PSH A ;SAVE LS BYTE OF RESULT
TIA ;MOVE MSBYTE OF DIFF INTO ACCA
AUC A TEMPA ;MSBYTE OF DIFF+ MSBYTE OF > # + CARRY BIT
EAA
PUL B ; RETRIEVE LS BYTE OF RESULT
TST CARRY ; ACCA, ACCB = RESULT OF BCD SUBTRACT
BEQ BACK ; IF NO OV NO NEED CLEAN STACK
INS ; CLEAN UP STACK
INS
BACK
RTS

"SHAENC" SUBROUTINE
READS SHAFT ANGLE ENCODERS

SHAENC LDA A MSBSAZ ; REPOS AZIMUTH ANGLE
LDA B LSBSAZ
STA A MBBENC ; STORES ANGLE IN TEMP LOC
STA B LSBENC
ASL #3 ; SCALE DAC OUTPUT BY FACTOR OF 2
ROL A
STA A DDBR2 ; OUTPUT MS 4 BITS OF AZ TO DAC
STA B DDBR2
LDA A MSBENC ; GET OLD A AND B
LDA B LSBENC
LDX #DIVISO
JSR DIVIDE ; DIVIDES ANGLE BY 14,912
LDX #XBCD
JSR BINBCD
PSH A
PSH B

ADDITION OF SHAENC SUBROUTINE - CK + AND - AZ LIMITS

STA B TEMPB ; SAVES ACCB TEMPO
LDA B SIGN ; START LIM CK
CMP B #2BH
BEQ PAL ; BR IF AZ COORD +
LDA B TEMPB ; AZ COORDS -
LDX #NAZLIM
JSR BCDSUB ; ACCX-NAZLIM
LDA A CARRY
BNE SHA2 ; BR IF ACCX>NAZLIM

PAL1 LDA B #OFFH ; POSTITIONER EXCEEDED LIMIT
STA B LFLAGA ; SET AZ LIMIT FLAG
LDX #MSG1I
JSR ASCDIS1 ; DISPLAY "ANGLE LIMIT EXCEEDED"
JSR ALSTOP ; BR TO STOP BOTH MOTORS
JMP #MSG1L ; WAIT 1 SEC AND GO CNTRL LP

PAL LDA B TEMPB ; CK FOR + AZ LIMIT
LDX #PAZLIM ; GET + AZ LIMIT
JSR BCDSUB ; ACCX-PAZLIM
LDA A CARRY
BNE PAL1

SHA2 LDA A SIGN
STA A AZSIGN
LDA A #041H
STA A LETA
LDA A #05AH
STA A LETB
PUL B
PUL A

LDX #ANGLE ; UPACK BCD ANGLE
JSR BCDDIS ; DISPLAYS PACKED BCD ON PANEL
LDX #DAZA
JSR ASC2 ; READS EL ANGLE
LDA A MSBSSEL
STA A MBBSEL ; STORE ANGLE TEMPORILY
ASL A

ROL A

STA A DDRA3 ; OUTPUT MS 4 BITS OF EL TO DAC

STA B DDRB3 ; OUTPUT LS 4 BITS OF EL TO DAC

LDA A MSBENC ; GET OLD A AND B

LDA B LSBENC

LDX #DIVISO

JSR DIVIDE ; DIVIDE ANGLE BY 14,912

LDX #ELBCD

JSR BINBCD ; RET PACKED BCD

PSH A

PSH B

STA B TEMPB ; HANDLES CHANGE IN COORD SYSTEM

LDA B SIGN
DISTRIBUTION LIST

Dr. N. A. Kheir
University of Alabama in Huntsville
School of Engineering
Huntsville, Alabama 35899

DRSMI-RDF
-R
-LP
-RPR
-RPT

US Army Material Systems Analysis Activity
ATTN: DRHSY-MP
Aberdeen Proving Ground, MD 21005