

NASA Contractor Report 178350

DATA ANALYSIS AND SOFTWARE SUPPORT  
FOR THE EARTH RADIATION BUDGET  
EXPERIMENT

W. Edmonds  
S. Natarajan

ST Systems Corporation (STX)  
28 Research Drive  
Hampton, VA 23666

Contract NAS1-17851

August 1987

(NASA-CR-178350) DATA ANALYSIS AND SOFTWARE SUPPORT FOR THE EARTH RADIATION BUDGET EXPERIMENT (ST Systems Corp.) 91 p Avail: NTIS HC A05/MF A01 CSCL 09B N87-26569  
G3/61 0087061  
Unclas



National Aeronautics and  
Space Administration

Langley Research Center  
Hampton, Virginia 23665-5225

## TABLE OF CONTENTS

	<u>Page</u>
Abstract.....	iii
SECTION 1 - INTRODUCTION.....	1-1
SECTION 2 - THE EARTH RADIATION BUDGET EXPERIMENT.....	2-1
SECTION 3 - SOFTWARE DEVELOPMENT.....	3-1
SECTION 4 - DATA REDUCTION AND ANALYSIS.....	4-1
APPENDIX A - SOCC.....	A-1
APPENDIX B - SCANCHK.....	B-1
APPENDIX C - ESMCAL.....	C-1
APPENDIX D - SOLCA.....	D-1

## Abstract

Computer programming and data analysis efforts were carried out under this contract in support of the Earth Radiation Budget Experiment (ERBE) at NASA/Langley. In this final report there will be a brief description of ERBE followed by sections describing software development and data analysis for both pre-launch and post-launch instrument data.

**PRECEDING PAGE BLANK NOT FILLED**

## SECTION 1 - INTRODUCTION

Computer programming and analysis efforts were carried out in support of the Earth Radiation Budget Experiment (ERBE). ERBE will be described in section 2 of this final report. Section 3 will contain descriptions of the programs developed under this contract with the procedures needed to run them and sample printer and plotter output. Listings of some of the major programs will be included in the appendix. Data reduction and analysis efforts will be described in section 4.

## SECTION 2 - ERBE

The Earth Radiation Budget Experiment is a three satellite experiment designed to provide global measurement of radiation reflected and emitted by the earth. Each satellite (ERBS, NOAA-9 and NOAA-10) carried into orbit a pair of instruments consisting of a scanner and nonscanner. ERBS (Earth Radiation Budget Satellite) was launched from Space Shuttle mission 41-G on October 5, 1984. ERBS is at an altitude of 610 km and an orbital inclination of  $57^{\circ}$ . NOAA-9 and NOAA-10 were placed into polar orbit at  $99^{\circ}$  and altitude of 812 km by Atlas launch vehicles.

The scanner instruments consist of three narrow field-of-view channels which scan the earth every 4 seconds. The short wave channel is designed to measure reflected solar energy whereas the long wave channel measures energy emitted in the infrared by the earth. The total channel measures both types of radiation and serves as a check on the other two channels. Data from the scanner instruments can be used to validate data acquired by the wide field-of-view nonscanner instruments through a process of integration. The scanner also provides a means of acquiring bi-directional data on reflected energy which can be useful in validating computer models of the process of energy reflection by the earth. Inflight calibration of the scanner is accomplished in two ways. Internal calibration utilizes the stimulus of the SWICS (Short Wave Internal Calibration Source) while solar calibration uses the sun's energy viewed through the MAM (Mirror Attenuator Mosaic).

The nonscanner instruments consist of four earth viewing channels and a solar monitor. Of the four channels, there are two medium field-of-view channels and two channels wide fields-of-view. Each set of two consists of a short wave and a total channel. Long wave determinations are made based on the difference between the total and short wave readings. The nonscanner instruments have inflight calibration capability which consists of a solar monitoring channel for solar calibration and a SWICS (Short Wave Instrument Calibration Source) which is viewed by the four earth channels for internal calibration.

### SECTION 3 - SOFTWARE DEVELOPMENT

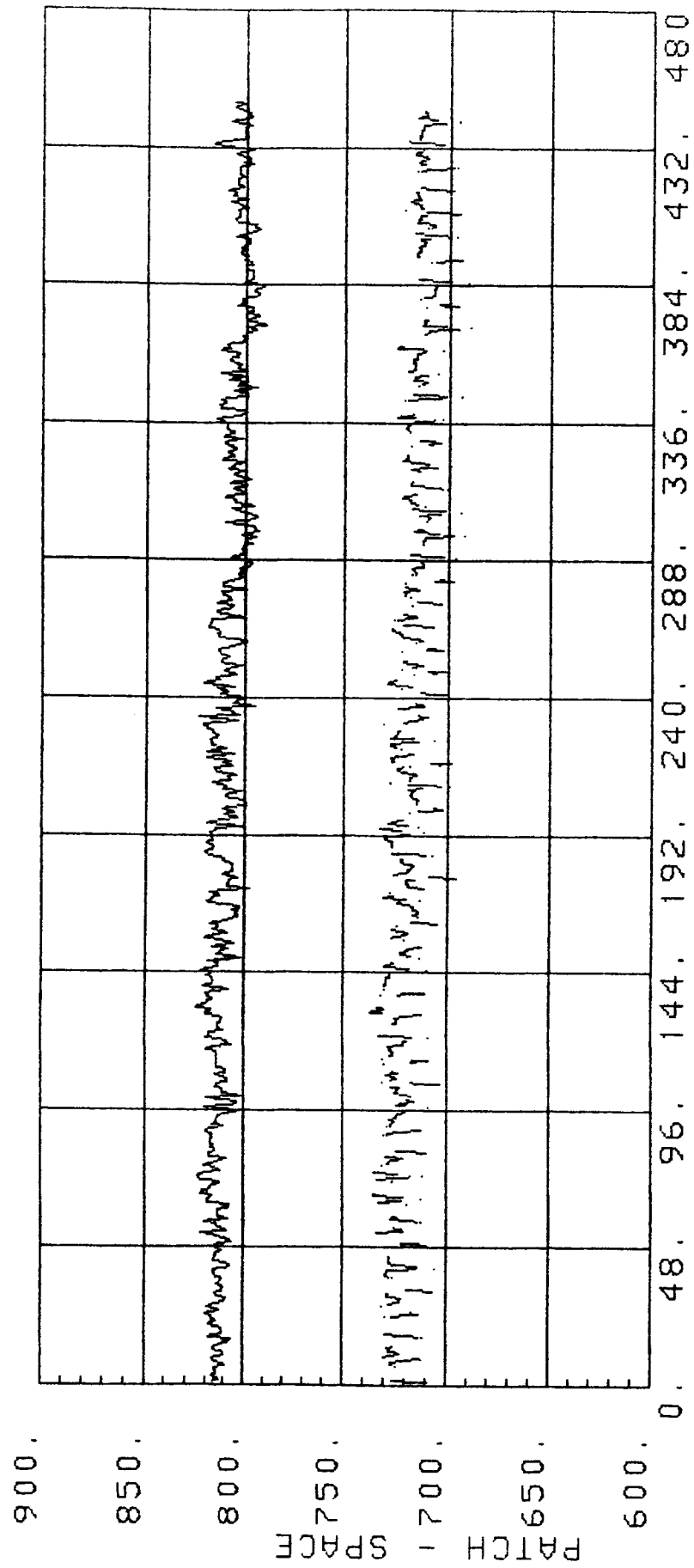
Software development under this contract was a continuation of efforts begun under a previous contract to support the ERBE project. Activities included both the writing of new programs on a variety of computer systems and the modification and updating of previously written software. Programs were developed to assist in the rapid data reduction of data acquired during satellite integration and during pre-launch check out at Vandenberg for NOAA-10. "ERTAB" was developed on the HP-1000 computer to generate a command echo tabulation which greatly assisted in finding data associated with special sequences and events such as: internal calibration, solar calibration, azimuth and elevation drive checks, heater checks and different scan modes (for the scanner instrument only).

"TICDL" (TIROS Internal Calibration Delta) and "ICDLT" (Internal Calibration Delta for ERBS) were developed on the HP-1000 to calculate the delta between the space look and internal calibration position data for the scanner instruments during their 4 levels (0, 1, 2 & 3) of calibration using the SWICS (Short Wave Internal Calibration Source). "BBPLT" was written to plot the calculated deltas involving scanner internal calibration data (see figure 1).

To facilitate processing of post launch data, several programs were written on the CDC computers to convert satellite data tapes into a format which could be processed by software previously developed on the HP-1000 computers. "NOACNVT" and

12:39:16.32 PM APR/13/85 \_\_\_\_\_SLCH

01:08:36.32 PM APR/13/85 \_\_\_\_\_STCH



SCANS

Figure 1



"PCMCNVT" were programs written on the NOS CDC computers to convert TIROS and ERBS data tapes to HP-1000 "TRW" format.

In addition to the tape conversion programs, other software was developed on NOS to process calibration data for scanner and nonscanner instruments on a regular basis. Considerable time was saved by submitting automatic batch jobs to process satellite data tapes as they arrived at NASA/Langley.

Among the other NOS programs developed were: "SOLCA", "ESCAN", "TSCAN", "SMCAL" and "ESMCAL".

"SOLCA" generates printouts and plots of nonscanner solar calibration data (see figure 2).

"ESCAN" and "TSCAN" extracted data for ERBS and TIROS scanner instruments for subsequent plotting by "ESMCAL" and "SMCAL" respectively.

A major software development task under this contract concerned the need to monitor the progress of the two sets of instruments on the NOAA-9 and NOAA-10 satellites on a real-time basis. Software was written to acquire, via dedicated telephone line, data from "SOCC" (Satellite Operations Control Center). The on-line real-time program ("SOCC") written in TURBO Pascal on the IBM-XT, was developed to display, limit check, and archive to disk all the ERBE data available during each pass of the NOAA satellite (see listing in appendix). Additional software ("REPLAY") was developed to play back the acquired data for quick review. One of the options included in "REPLAY" is the capability to save to disk snapshots of the data displayed on the CRT. These snapshots are used to generate plots and tabulations

NOAA-9 NONSCANNER SOLAR CALIBRATION

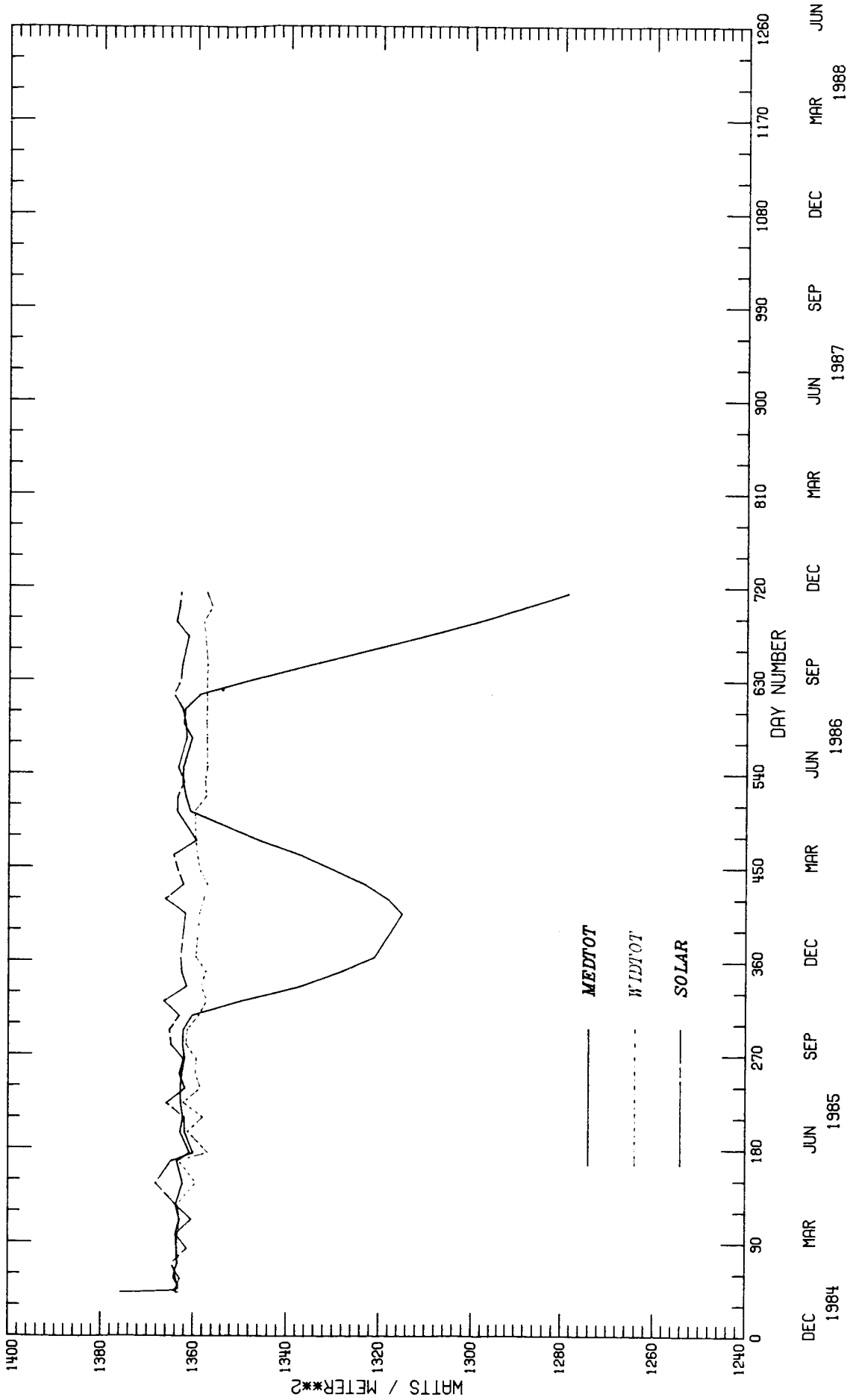


Figure 2

of parameters of interest. A program called "SCANCHK" (scan-check) was also developed on the IBM-XT to tabulate scanner position data. This program has been very useful in diagnosing problems with both NOAA-9 and NOAA-10 scan position data.

## SECTION 4 - DATA REDUCTION AND ANALYSIS

Data reduction and analysis support under this contract consisted of NOAA-10 pre-launch test support and post-launch data analysis support for all three satellites: ERBS, NOAA-9 and NOAA-10.

During pre-launch testing of the ERBE instruments, support activities included data tape copying and reformatting and SEPET (Spacecraft Electronic Performance and Evaluation Test) data analysis. Data acquired from the NOAA-10 spacecraft was written to magnetic tape in "TIP" (TIROS Information Processor) format. In order to quickly determine the performance of the ERBE instruments, it was necessary to convert these data tapes to a format compatible with software previously developed on an HP-1000 computer.

After reformatting, data from the SEPET tests was analyzed, plotted and tabulated. Internal calibration, solar calibration, azimuth and elevation drive checks were evaluated for both scanner and nonscanner instruments. For the scanner instrument, all five scan modes (nadir, short, normal, mam and stow) were examined. For the nonscanner, all three levels of the SWICS (Short Wave Internal Calibration Source) output were plotted. Command echoes for both instruments were tabulated to verify proper commanding and execution of the various test sequences. The SEPET test analysis activities were performed numerous times at the RCA facility in Hightstown, NJ (RCA was the prime NOAA-10 satellite contractor) and also at Vandenberg AFB in California.

Post launch analysis support was given for all three satellites carrying ERBE instruments (NOAA-9, NOAA-10 and ERBS).

Routine data reduction consisted of examining internal calibration and solar calibration data for scanner and nonscanner instruments. When the scanner on ERBS began to have difficulty scanning (around May 1985), closer examination of the data was necessary. Scan position data and housekeeping data were tabulated for considerable periods of time in an effort to determine the cause and effects of the scanner problem.

In carrying out the post launch data analysis, both the NOS facility at Langley and the HP-1000 computer were utilized. Software developed under this contract was used to convert data tapes in "LaRC" format to "TRW" format for processing on the HP-1000. This conversion facilitated the use of considerable engineering analysis software previously developed on the HP-1000 (both by STX personnel and TRW personnel) under previous contracts. The more routine data analysis was carried out in production mode on the NOS CDC computers.

## APPENDIX A - SOCC

Program SOCC inputs ERBE/TIROS data from the ADCCP BUFFER, processes and limit checks the data and displays it on a color monitor. The program checks the identity of the source of the data (which satellite) and saves NOAA-9 and NOAA-10 (ERBE) data in separate disk files for later review using the REPLAY program.

Listing of: SOCC.PAS

```

1  PROGRAM SOCC; { DISPLAY PROGRAM FOR ERBE/NOAA-9 & NOAA-10 }
2  {
4  written by :   William L. Edmonds, STX Corporation
5                December 1984
7  modified   :   Summer 1986 for NOAA-10 capability.
9  Program SOCC inputs ERBE/TIROS data from the ADCCP BUFFER, processes
10 and limitchecks the data and displays it on a color monitor. The
11 program checks the identity of the source of the data (which satellite)
12 and saves NOAA-9 and NOAA-10 (ERBE) data in separate disk files
13 for later review using the REPLAY program.   }

17 LABEL REEP ,stop,bottom;
18 TYPE ABC = STRING[80] ;
19 TYPE BYTEBUFF = ARRAY[0..8000] OF BYTE ;
20     { used for display memory...see var FRAME below}

22 TYPE integBUFF = ARRAY[0..4000] OF INTEGER ;
23 TYPE BITE = ARRAY[0..80] OF BYTE ;
24 TYPE MinorFrameBufs = Array[0..103] of byte ;
25 TYPE dualbufs = Array[0..4] of MinorFrameBufs ;
26 TYPE MinorFramewords = Array[0..51] OF INTEGER ;
27 TYPE dualwordbufs = Array[0..4] of MinorFramewords ;
28 TYPE ERBREC = ARRAY[0..21] OF BYTE ;
29     { USED IN SAVING ERBE DATA TO DISK }

31 TYPE REGPACK = RECORD
32     ax,bx,cx,dx,bp,di,si,ds,es,flags:integer;
33     END;
34 TYPE COMBUFF = ARRAY[0..519] OF BYTE ;
35     {BUFFER FOR ADCCP COMMUNICATIONS}

37 type smallbufw = array[0..3] of integer ;
38 TYPE COMBUFFW = ARRAY[0..259] OF INTEGER ; {WORD BUFFER FOR ADCCP}
39 CONST
40     hexdig: array[0..15] of char = '0123456789abcdef' ;
41     { used in dec to hex conversion }

43     ON : STRING[3] = 'ON' ;{USED FOR DIGB DATA}
44     OFF : STRING[3] = 'OFF' ;{USED FOR DIGB DATA}
45     NOAA9 : STRING[10] = 'NOAA9.DAT' ;
46     NOAA10 : STRING[10] = 'NOAA10.DAT' ;
47     WHITE: INTEGER = 15 ; { color display attribute values }
48     RED: INTEGER = 12 ;
49     YELLOW : INTEGER = 14 ;
50     GREEN : INTEGER = 10 ;

```

## Listing of: SOCC.PAS

```

51     msgrcv : byte = 2 ;      { ADCCP protocol: message received }
52     xmtrcv : byte = 7 ;      { ADCCP protocol: command to send or receive }
53     maxsec : integer = 10 ;   { ADCCP protocol: time out value }
54     sync : byte = $32 ;       { ADCCP protocol: sync byte value }
55     bisync : integer = $220 ; { hex address of ADCCP bisync port }
56     noerr : integer = 0 ;     { ADCCP code for no error }
57     ptlerr : integer = 1 ;    { ADCCP code for protocol error }
58     cserr : integer = 2 ;     { ADCCP code for checksum error }
59     timerr : integer = 3 ;    { ADCCP code for time out error }
60     id0msg : array[0..3] of byte = ($32,$32,0,$8) ; { idle message # 0 }
61     id1msg : array[0..3] of byte = ($32,$32,0,$c) ; { idle message # 1 }
62     idmsln : integer = 4 ;    { idle message length in bytes }
63 VAR
64     ES IPL : ABC ; { SCANNER PULSE-LOAD BUS INDICATOR = A OR B }
65     ERBDAT : ERBREC ; { THIS ARRAY HOLDS ERBE DATA TO BE SAVED TO DISK }
66     F9 : FILE OF ERBREC ; { Disk file used to save ERBE/NOAA-9 data }
67     F10 : FILE OF ERBREC ; { ERBE/NOAA-10 DATA FILE }
68     key : integer ; { holds value of last key hit on keyboard }
69     state : byte ; { ADCCP protocol : communications state }
70     SCID : BYTE ;
71         { SCID SHOULD BE = HEX "D" or "E" for NOAA-9,
72           "F" or "0" FOR NOAA-10 }

74     OLDSCID : BYTE ; { USED TO CHECK FOR CHANGE IN SCID }
75     FORG : string[2] ; { NOAA- " 9 or 10", (f or g) ie. "FORG" }
76     id : byte ; { ADCCP idle message id: either 0 or 1 }
77     msgid : byte ; { ADCCP protocol: message id 0 thru 3 }
78     elpmin : byte ; { elapsed minutes calculated by INITRCV-- ADCCP }
79     elpsec : byte ; { elapsed seconds calculated by INITRCV }

81     RECPACK : REGPACK ; { register pack used during interrupt processing
82     ah,al,ch,cl,dh : byte ; { register names high and low }
83     TIME : ARRAY[0..5] OF BYTE ; { holds major frame time ZULU }
84     MN : INTEGER ; { minor frame # within buffer: 0 to 4 }
85     status,count: integer ;
86     linenum,charnum: integer ; { used by DUMPSCRN }
87     MinorFrameNUM : integer ; { ERBE minorframe #: 0 to 319 }
88     MajorFrameNUM : integer ; { ERBE major frame # }
89     Digb : byte ; { DIGITAL "B" byte from NOAA data stream }
90     scrnmde: array[0..15] of byte ;
91         { holds display parameters for CONOGRAPHICS }

93     SDIGA : ARRAY[0..7] OF INTEGER ;
94         { scanner digital "A" data from 1 NOAA minor frame }

96     NDIGA: INTEGER ; { non-scanner digital "A" data for 1 minorframe }
97     ANALOG : byte ; { analog byte from 1 NOAA minor frame }
98     RANALOG : REAL ;
99         { analog converted to real decimal or non-integer value }

101    VALST : ABC ;

```



Listing of: SOCC.PAS

```

102     LENG : INTEGER ;
103         { length in bytes of ADCCP message received: full = 520 }

105     RCVBUF : COMBUFF ABSOLUTE $8F80: $0000 ;
106         { absolute addr of receive buffer }

108     RCVBUFW : COMBUFFW ABSOLUTE $8F80:$0004 ;
109     msgbuf : smallbufw absolute $8F80: $0000 ;
110     XMTBUF : COMBUFF ABSOLUTE $8F80: $0400 ;
111     MinorFrame : dualbuffs absolute $7f80 : $0004 ; { 9800:0004 ; }
112     MinorFrameW : dualwordbuffs absolute $7f80 : $0004 ; { 9800:0004 ; }
113     MINFW : COMBUFFW ABSOLUTE $7f80 : $0004 ; { 9800:0004 ; }
114     FRAME: integBUFF ABSOLUTE $B800:$0000;
115     BFRAME: BYTEBUFF ABSOLUTE $B800:$0000;
116     txt : text ;
117     txtfile : string[10] ;
118     SORC:ABC ; {TEXT STRING USED FOR CONOGRAPHICS DISPLAY}
119     ATT, XY: INTEGER; I,K, J:INTEGER;
120     SCRINT: ARRAY[0..1] OF INTEGER ABSOLUTE $0000:$0014;
121     STORINT : ARRAY[0..1] OF INTEGER ;
122     statpr : byte absolute $0050:$0000;
123         {INTERRUPT VECTOR LOCATION FOR PRINTING}

128     {-----scanner digital "a" variables-----}
129     TYPE COUNTS = INTEGER;
130     VAR
131     SSCH,SLCH,STCH : COUNTS ; { CHANNEL OUTPUT }
132     SSWCSA,SSWCST : REAL ; { SWICS AMP OUTPUT & TEMP }
133     SSDACV,SLDACV,STDACV : REAL ; { DAC VOLTAGES }
134     SDPBV,SDNBV : REAL ; { POS & NEG DETECTOR BIAS VOLTAGES}
135     STRV1,STRV2 : REAL ; { TEMP REF VOLTAGES}
136     SSDETT,SLDETT,STDETT : REAL ; { DETECTOR TEMPS}
137     SLBBT,STBBT : REAL ; { BLACKBODY TEMPS }
138     SSMBT,STMBT : REAL ; { MAM BAFFLE TEMPS }
139     SSMAMT,STMAMT : REAL ; { MAM TEMPS }
140     SCMDE,SSTAT,SCPOS : COUNTS ; { COMMAND ECHO, STATUS, SCAN POSITION}
141     SAPOSL,SAPOSH: BYTE ; { LOW & HIGH AZIMUTH POSITION BYTES }
142     SAZP : REAL ; { AZIMUTH POSITION }
143     SAPOS : COUNTS ;
144     TREF : REAL ; { TEMP REAL NUM }

146     { -----EXTERNALS-----}

150     PROCEDURE S5080(var i :byte); EXTERNAL 'CONO.COM';
151     { S5080 PUTS THE CONOGRAPHICS SYSTEM IN THE DESIRED MODE:
152     At program start, it puts the screen in 50 row,80 column mode.

```

Listing of: SOCC.PAS

```

153     At termination, it returns the screen to 25 X 80 . )

155  PROCEDURE PUTOUT(VAR SORC:ABC;VAR FRAME:INTEGER;ATTR:INTEGER);
156  EXTERNAL 'PUTOUT.COM';
157  { PUTOUT places a string and its color
158     attributes in the screen memory area }

160  FUNCTION PRSTAT:INTEGER; EXTERNAL 'PRSTAT.COM';
161  { PRSTAT responds to the shift-PrtSC keys by setting a flag.
162    The program will then dump the screen to the printer
163    50 rows by 80 columns }

165  { -----PROCEDURES & FUNCTIONS-----}

167  PROCEDURE OUTPUT(VAR SORC: ABC; VAR FRAME: INTEGER; ATTR: INTEGER);
168  { OUTPUT WAS INTENDED TO SOLVE THE PROBLEM OF GARBAGE DIGITS LEFT ON
169    THE CONOGRAPHICS SCREEN WHEN A LONG STRING OF NUMBERS WAS REPLACED
170    BY A SHORT STRING. THIS PROBLEM WAS NEVER SOLVED DUE TO LACK OF
171    TIME. THE IDEA WAS TO WRITE OUT BLANKS FIRST AND THEN PUT THE
172    DESIRED NUMBER OUT WITH THE PUTOUT ROUTINE ABOVE }
173  VAR BLANKS : ABC ;
174  BEGIN
175  {      BLANKS := '          ' ; 10 BLANKS }
176  {      PUTOUT(BLANKS,FRAME,WHITE);    }
177  PUTOUT(SORC,FRAME,ATTR);
178  END;

180  function GETKEY : integer ; {GET VALUE OF KEY STRUCK...
181                             IF NO KEY THEN ZERO IS RETURNED }
182  begin
183      with recpack do {SET UP FOR INTERRUPT 21 HEX WITH AH = 6
184                      AND DX = FF HEX }
185          begin
186              ah := 6 ;
187              al := 0;
188              ax := ah shl 8 + al ;
189              dx := $ff ;
190          end;
191          intr($21,recpack); { DO INTERRUPT 21 }
192          with recpack do
193              begin
194                  al := ax and $ff ; { GET VALUE OF CHARACTER }
195              end;
196          GETKEY := al ; { IF NO KEY WAS HIT, ZERO WILL BE RETURNED}
197  end;

203  function CHKTIM : boolean ;

```

Listing of: SOCC.PAS

```

204           { TIMING ROUTINE FOR ADCCP COMMUNICATIONS}
205   BEGIN
206       AH := $2C; { SET UP FOR INTERRUPT 21 HEX WITH AH = 2C HEX}
207       WITH RECPACK DO
208           BEGIN
209               AX := AH SHL 8 ;
210           END;
211       INTR($21,RECPACK);
212       WITH RECPACK DO { EXTRACT THE TIME }
213           BEGIN
214               DH := DX SHR 8 ;
215               IF ( dh < elpsec ) then
216                   begin
217                       CL := CX AND 255 ;
218                       cl := cl -1 ;
219                       dh := dh +60 ;
220                   end;
221               dh := dh - elpsec ;
222               if ( dh < maxsec ) then
223                   begin
224                       if ( cl <> elpmin ) then chktim := true
225                       else chktim := false ;
226                   end
227               else chktim := true ;
228           END;

230   END;

232   PROCEDURE SENDIDLE ;
233       { ADCCP ROUTINE TO SEND APPROPRIATE IDLE MESSAGE }

235   VAR I : INTEGER ;
236   BEGIN
237       CASE ID OF

239           0:   FOR I := 0 TO 3 DO
240               XMTBUF[I] := IDOMSG[I] ;

242           1:   FOR I := 0 TO 3 DO
243               XMTBUF[I] := ID1MSG[I] ;
244           else i:= i ;
245           END; { OF CASE }
246           PORT[BISYNC] := XMTRCV ;
247           AH := $2C;
248           WITH RECPACK DO
249               BEGIN
250                   AX := AH SHL 8 ;
251               END;
252           INTR($21,RECPACK);
253           WITH RECPACK DO
254               BEGIN

```

## Listing of: SOCC.PAS

```

255         DH := DX SHR 8 ;
256         CL := CX AND 255 ;
257         ELPSEC := DH ;
258         ELPMIN := CL ;
259     END;
260 END; { OF SENDIDLE }

262 FUNCTION XYPOS(ROW,COL:INTEGER ):INTEGER ;
263     { CALCULATE LINEAR ARRAY POSITION
264     FOR CONOGRAPHICS DATA TO BE DISPLAYED AT ROW,COL }
265 BEGIN
266     XYPOS := ROW * 80 + COL;
267 END;

269 PROCEDURE XFER;
270     { ADCCP COMMUNICATIONS ROUTINE TO TRANSFER DATA FROM THE
271     RECEIVE BUFFER TO AN ARRAY FOR PROCESSING }
272 VAR I : INTEGER ;
273 BEGIN
274     LENG := (msgBUF[1] AND $3FF) ;
275     str(leng:5,sorc);
276     putout(sorc,frame[xypos(46,18)],green);
277     FOR I := 0 TO 519 DO
278         MINFW[1] := RCVBUF[1] ;
279     END; { OF XFER }

281 PROCEDURE INITRCV ; { INITIATE COMMUNICATIONS WITH ADCCP BOX }
282 LABEL WAIT, MESSRCV,XIT ;

284 BEGIN
285     ID := 0 ;
286 WAIT: SENDIDLE ;
287     STATUS := PORT[BISYNC] AND MSGRCV ;
288     IF STATUS <> 0 THEN GOTO MESSRCV ;
289     IF NOT(CHKTIM) THEN GOTO WAIT ;
290     STATUS := TIMERR ;
291     GOTO XIT ;

293 MESSRCV: IF RCVBUF[0] <> 0 THEN
294     BEGIN
295         STATUS := CSERR ;
296         GOTO XIT ;
297     END
298     ELSE
299     BEGIN
300         MSGID := RCVBUF[3] AND 4 ;
301         IF MSGID <> 0 THEN
302             BEGIN
303                 STATUS := PTLERR ;
304                 GOTO XIT ;
305             END;

```

## Listing of: SOCC.PAS

```

306         END;
307         ID := 1;
308         SENDIDLE ;
309         STATE := 1;
310         STATUS := NOERR ;

312  XIT: END;

315  PROCEDURE GETBUF; { GET A BUFFER OF DATA FROM ADCCP }
316  LABEL GOTMSG,AGIN,DONE,gdms00,gdmsg ;
317  BEGIN

319  AGIN: STATUS := PORT[BISYNC] AND MSGRCV ;
320         IF STATUS <> 0 THEN GOTO GOTMSG ;
321         IF (NOT(CHKTIM)) THEN GOTO AGIN ;
322         SENDIDLE;
323         STATUS := TIMERR;
324         GOTO DONE;
325  GOTMSG: IF (RCVBUF[0] <> 0 )THEN
326         BEGIN
327             SENDIDLE;
328             STATUS := CSERR;
329             GOTO DONE ;
330         END;
331         MSGID := (RCVBUF[3] AND 4) SHR 2 ;
332         IF (MSGID = STATE) THEN goto gdmsg ;
333         if ( state = 1 ) then goto gdms00 ;
334         BEGIN
335             SENDIDLE ;
336             STATUS := PTLERR;
337             GOTO DONE;
338         END;
339  gdmsg:     STATE := (NOT(STATE)AND 1) ;
340         ID := STATE ;
341  gdms00:   XFER;
342         SENDIDLE;
343         STATUS := NOERR;

345  DONE: END;

347  FUNCTION ONOFF(DIGB,I:BYTE):ABC ; { USED IN DISPLAY OF DIGB DATA }
348  BEGIN
349         ONOFF := ON ;
350         IF((DIGB AND I)>0 ) THEN ONOFF := OFF ;
351  END;

355  procedure NEWSCREEN ; { SET UP CONOGRAPHICS FOR 80 COL BY 50 ROWS }
356  BEGIN

```

## Listing of: SOCC.PAS

```

357     SCRNMODE[0] := #71;
358     SCRNMODE[1] := #50;
359     SCRNMODE[2] := #5A;
360     SCRNMODE[3] := #0F;
361     SCRNMODE[4] := #1B;
362     SCRNMODE[5] := 6;
363     SCRNMODE[6] := #19;
364     SCRNMODE[7] := #1A;
365     SCRNMODE[8] := 3;
366     SCRNMODE[9] := 7;
367     SCRNMODE[10] := #20 ;
368     SCRNMODE[11] := #20 ;
369     SCRNMODE[12] := 0;
370     SCRNMODE[13] := 0;
371     SCRNMODE[14] := 0;
372     SCRNMODE[15] := 0;
373     S5080(SCRNMODE[0]); { CALL ROUTINE TO SEND DATA TO CONOGRAPHICS }

375     END;

377     PROCEDURE OLDSCREEN ; { RESTORES SCREEN TO NORMAL MODE }
378     VAR LOC : INTEGER ;
379     BEGIN
380         FOR LOC := 0 TO 3999 DO
381             FRAME[LOC] := #F00 ;

383             SCRNMODE[4] := #1F ;
384             SCRNMODE[7] := #1C ;
385             SCRNMODE[8] := 2;
386             SCRNMODE[10] := 6;
387             SCRNMODE[11] := 7;
388             S5080(SCRNMODE[0]);
389     END;

392     PROCEDURE LIMITCHECK(X,RL,YL,YH,RH:REAL;VAR SORC:ABC;VAR ATT:INTEGER);
393     { LIMITCHECK DETERMINES WHAT COLOR TO DISPLAY A PARAMETER IN AND
394     APPENDS TO THE STRING "SORC" THE APPROPRIATE SUFFIX RL,YL,YH,RH
395     DEPENDING ON THE RED LOW, YELLOW LOW ETC SITUATION}

397     BEGIN
398         ATT := GREEN ;
399         IF X < YL THEN
400             BEGIN
401                 IF X < RL THEN
402                     BEGIN
403                         ATT := RED ;
404                         SORC := SORC + 'RL' ;
405                     END
406                 ELSE
407                     BEGIN

```

## Listing of: SOCC.PAS

```

408             ATT := YELLOW ;
409             SORC := SORC + 'YL' ;
410         END;
411     END
412     ELSE
413     BEGIN
414     IF X > YH THEN
415     BEGIN
416         IF X > RH THEN
417         BEGIN
418             ATT := RED ;
419             SORC := SORC + 'RH' ;
420         END
421         ELSE
422         BEGIN
423             ATT := YELLOW ;
424             SORC := SORC + 'YH' ;
425         END;
426     END
427     ELSE SORC := SORC + '    ' ;
428     END;
429 END; (OF LIMITCHECK )

432 PROCEDURE DisplayTime;
433     { DISPLAY THE TIME VALUE EXTRACTED FROM THE INCOMING DATA }

435 VAR TIMSTRING : ABC ;
436 DAYS,HRS,MINS : INTEGER;  millisecs,SECS : REAL ;
437 BEGIN
438     days := (time[0]shr 1) +((time[1] and 128)shr 7) ;
439     millisecs := ((( time[1]and 7)*256.0 + time[2])*256.0 + time[3])
440                 *256.0 +time[4];
441     hrs := trunc(millisecs/3600000.0) ;
442     mins := trunc(millisecs/60000.0) mod 60 ;
443     secs := trunc((millisecs/1000.0)-mins*60.0-hrs*3600.0);
444     str(days:4,sorc); putout(sorc,frame[xypos(42,40)],white);
445     str(hrs:2,sorc); putout(sorc,frame[xypos(42,45)],white);
446     str(mins:2,sorc); putout(sorc,frame[xypos(42,50)],white);
447     str(secs:6:3,sorc); putout(sorc,frame[xypos(42,55)], white);
448 END;

451 PROCEDURE DISPLAYACRO ;
452     { DISPLAY THE TEMPLATE OF ACRONYMS ON THE CONOGRAPHICS SCREEN }

454 VAR I: INTEGER;
455 BEGIN
456     txtfile := 'sorc.txt' ;
457     assign(txt,txtfile);
458     reset(txt);

```

## Listing of: SOCC.PAS

```

459   att := 15 ;
460   i := 80 ;
461   while not eof(txt) do
462   begin
463   readln(txt,sorc);
464   sorc := sorc + '          ' ;
465   putout(sorc,frame[i],att);
466   i := i + 80 ;
467   end;
468   close (txt);

470   END;

473   PROCEDURE DIVY(N:INTEGER);
474       { EXTRACTS USABLE DATA FROM TIP MINOR FRAME N }

476   LABEL RETURN ;
477   var nextmf,tipstatus: integer ;   nmf,mf: string[4] ;
478   BEGIN
479       { CALCULATE EXPECTED MINOR FRAME NUMBER AND CHECK THE ACTUAL
480       MINOR FRAME NUMBER RECEIVED. IF NOT EQUAL, DISPLAY DIAGNOSTIC
481       MESSAGE }
482       nextmf := minorframenum +1 ;
483       if (nextmf > 319 ) then nextmf := 0 ;
484       MINORFRAMENUM := MINORFRAME[N,5] + (MINORFRAME[N,4]AND 1) shl 8;
485       if (minorframenum<>nextmf) then
486       begin
487           str(nextmf:4,nmf);
488           str(minorframenum:4,mf);
489           sorc := 'expecting mf ' + nmf + ' but found mf ' + mf ;
490           putout(sorc,frame[xypos(45,2)],yellow);
491       end ;
492       { CHECK SPACE CRAFT ID TO DETERMINE NOAA-9,NOAA-10 OR OTHER
493       NOAA SATELLITE. WE ARE INTERESTED ONLY IN NOAA-9 & 10 }
494       SCID := MINORFRAME[N,2] AND 15 ;
495       IF ((SCID <> $0D)AND(SCID<>$0F)AND(SCID<>$00)
496       AND(SCID<>$0E)) THEN GOTO RETURN ;
497       {
498           IF SATELLITE ID IS NEITHER NOAA-F NOR NOAA-G
499       }
500       { NOAA-9 CAN HAVE EITHER A HEX '0D' OR '0E' ID }
501       { NOAA-10 CAN HAVE EITHER A HEX '0F' OR '00' ID }
502       IF (( SCID = $0D) OR(SCID = $0E)) THEN FORG := ' 9'
503       ELSE
504       FORG := '10' ;
505       { IF THE ID HAS CHANGED SINCE THE LAST FRAME OF DATA,
506       CLEAR THE SCREEN AND START WITH A NEW TEMPLATE }
507       IF (SCID <> OLDSCID ) THEN DISPLAYACRO ;
508       OLDSCID := SCID ;
509       sorc := forg ;

```



## Listing of: SOCC.PAS

```

510      putout(sorc,frame[xypos(18,40)],green);
511      if (minorframenum = 0) then
512      begin
513          sorc := '
514          putout(sorc,frame[xypos(45,2)],yellow);
515          FOR I := 0 TO 4 DO
516              TIME[I] := MINORFRAME[N,I+8] ;
517          displaytime;
518      END;
519      TIPSTATUS := (MINORFRAME[N,3] AND 96) SHR 5 ;
520      ATT := RED ; ( DEFAULT ATTRIBUTE COLOR, ONLY ORBIT MODE IS GREEN )
521      CASE TIPSTATUS OF

523  0:  BEGIN
524          SORC := 'ORBIT MODE      ' ;
525          ATT := GREEN ;
526      END;
527  1:  SORC := 'DUMP MODE      ' ;
528  2:  SORC := 'DWELL MODE    ' ;
529  3:  SORC := 'UNDEFINED MODE' ;
530  else sorc := 'undefined mode' ;
531      END;
532      PUTOUT(SORC,FRAME[XYPOS(41,2)],ATT);
533  IF (MINORFRAMENUM > 319) THEN GOTO RETURN ;
534      DIGB := MINORFRAME[N,12];
535      ANALOG := MINORFRAME[N,13] ;
536      NDIGA := (MINORFRAME[N,52] shl 8) or minorframe[n,53] ;
537      SDIGA[0] := (MINORFRAME[N,18] shl 8 or minorframe[n,19] ) SHR 4 ;
538      SDIGA[1] :=(((MINORFRAME[N,19] AND 15) SHL 8) OR MINORFRAME[N,28]);
539      SDIGA[2] := (MINORFRAME[N,29] SHL 4) OR ( MINORFRAME[N,44] SHR 4) ;
540      SDIGA[3] := ((MINORFRAME[N,44] and 15)shl 8 or minorframe[n,45] ) ;
541      SDIGA[4] := (MINORFRAME[N,60] shl 8 or minorframe[n,61] ) SHR 4;
542      SDIGA[5] :=(((MINORFRAME[N,61] AND 15 ) SHL 8) OR MINORFRAME[N,72]);
543      SDIGA[6] := (MINORFRAME[N,73] SHL 4) OR (MINORFRAME[N,86] SHR 4 ) ;
544      SDIGA[7] :=(( MINORFRAME[N,86] and 15) shl 8 or minorframe[n,87] ) ;
545  ( ERBDAT ARRAY IS USED TO SAVE THE RAW VALUES OF THE ERBE DATA FROM THE
546  NOAA 'TIF' DATA STREAM. OUT OF EACH 520 BYTE BUFFER RECEIVED FROM
547  THE ADCCP BOX, 5 MINOR FRAMES OF DATA WITH 22 BYTES OF USEFUL INFO
548  EACH IS SAVED TO DISK )
549      ERBDAT[0] := (MINORFRAME[N,4] AND 1) OR (SCID SHL 1) ;
550      ERBDAT[1] := MINORFRAME[N,5] ;
551      ERBDAT[2] := MINORFRAME[N,8];
552      ERBDAT[3] := MINORFRAME[N,9];
553      ERBDAT[4] := MINORFRAME[N,10] ;
554      ERBDAT[5] := MINORFRAME[N,11] ;
555      ERBDAT[6] := MINORFRAME[N,12] ;
556      ERBDAT[7] := MINORFRAME[N,13] ;
557      ERBDAT[8] := MINORFRAME[N,18] ;
558      ERBDAT[9] := MINORFRAME[N,19] ;
559      ERBDAT[10] := MINORFRAME[N,28] ;
560      ERBDAT[11] := MINORFRAME[N,29] ;

```

Listing of: SOCC.PAS

```

561     ERBDAT[12] := MINORFRAME[N,44] ;
562     ERBDAT[13] := MINORFRAME[N,45] ;
563     ERBDAT[14] := MINORFRAME[N,52] ;
564     ERBDAT[15] := MINORFRAME[N,53] ;
565     ERBDAT[16] := MINORFRAME[N,60] ;
566     ERBDAT[17] := MINORFRAME[N,61] ;
567     ERBDAT[18] := MINORFRAME[N,72] ;
568     ERBDAT[19] := MINORFRAME[N,73] ;
569     ERBDAT[20] := MINORFRAME[N,86] ;
570     ERBDAT[21] := MINORFRAME[N,87] ;
571     IF (FORG = '9') THEN WRITE(F9,ERBDAT)
572     ELSE WRITE(F10,ERBDAT) ;
573         ssch := sdiga[0] ;
574         slch := sdiga[1] ;
575         stch := sdiga[2] ;
576         scpos := sdiga[3] ;
577     RETURN;
578     END; { OF DIVY }

581     PROCEDURE PROCESSDIGB ;
582         { PROCESS DIGITAL 'B' DATA FOR SCANNER & NONSCANNER}

584     BEGIN
585         CASE MINORFRAMENUM MOD 32 OF

587             0: BEGIN           { SCAN MOTOR POWER ON=0, OFF=1 }
588                 { EXCEPT NOT AVAIL IN MINORFRAME 0 }

590                 IF MINORFRAMENUM>0 THEN
591                     BEGIN
592                         SORC := ONOFF(DIGB,16) ;
593                         PUTOUT(SORC,FRAME[XYPOS(11,14)],GREEN);
594                     END;
595                 END;
596             3: BEGIN           { SCANNER PULSE LOAD BUS A }
597                 SORC := ONOFF(DIGB,64);
598                 IF(SORC = ON ) THEN ES IPL := ' A' ELSE IF (ESIPL = ' A')
599                 THEN ES IPL := OFF ;
600                 PUTOUT(ESIPL,FRAME[XYPOS(12,54)],GREEN);

602                 { SCANNER BLACKBODY HEATER POWER }
603                 SORC := ONOFF(DIGB,32);
604                 PUTOUT(SORC,FRAME[XYPOS(15,54)],GREEN);

606                 { NON- SCANNER BLACKBODY HEATER POWER }
607                 SORC := ONOFF(DIGB,16);
608                 PUTOUT(SORC,FRAME[XYPOS(33,12)],GREEN);
609                 END;

611             4: BEGIN           { NON-SCANNER AZIMUTH MOTOR POWER }

```

## Listing of: SOCC.PAS

```

612      SORC := ONOFF(DIGB,128);
613      PUTOUT(SORC,FRAME[XYPOS(33,34)],GREEN);

615      { NON-SCANNER SPARE WOULD GO HERE ALSO }
616      END;
617      8: BEGIN          { SCANNER PED STANDBY HEATER }
618          SORC := ONOFF(DIGB,16);
619          PUTOUT(SORC,FRAME[XYPOS(14,54)],GREEN);
620          END;
621      10: BEGIN         { NON-SCANNER INSTRUMENT HEATER PWR }
622          SORC := ONOFF(DIGB,64);
623          PUTOUT(SORC,FRAME[XYPOS(34,12)],GREEN);
624          END;
625      11: BEGIN         { NON-SCANNER ELEVATION MOTOR POWER }
626          SORC := ONOFF(DIGB,128);
627          PUTOUT(SORC,FRAME[XYPOS(31,34)],GREEN);
628          END;
629      13: BEGIN         { SCANNER PULSE LOAD BUS B }
630          SORC := ONOFF(DIGB,64);
631          IF(SORC = ON )
632          THEN ESIPL := ' B'
633          ELSE IF (ESIPL = ' B' )
634              THEN
635                  ESIPL := OFF ;
636          PUTOUT(ESIPL,FRAME[XYPOS(12,54)],GREEN);
637          END;
638      17: BEGIN         { SCANNER INSTR POWER }
639          SORC := ONOFF(DIGB,128);
640          PUTOUT(SORC,FRAME[XYPOS(10,54)],GREEN);

642      { SCANNER STANDBY HEATER POWER}
643      SORC := ONOFF(DIGB,64);
644      PUTOUT(SORC,FRAME[XYPOS(16,54)],GREEN);

646      { SCANNER AZIMUTH MOTOR POWER}
647      SORC := ONOFF(DIGB,32);
648      PUTOUT(SORC,FRAME[XYPOS(13,34)],GREEN);

650      { SCANNER SPARE WOULD ALSO GO HERE }
651      END;
652      18: BEGIN         {NON-SCANNER INSTRUMENT POWER}
653          SORC := ONOFF(DIGB,128);
654          PUTOUT(SORC,FRAME[XYPOS(32,54)],GREEN);

656      {NON-SCANNER PULSE LOAD BUS A OR B}
657      SORC := ONOFF(DIGB,64) ;
658      IF( SORC = ' ON' ) THEN SORC := ' A ' ;
659      IF (ONOFF(DIGB,32) = ' ON' ) THEN SORC := ' B ' ;
660      PUTOUT(SORC,FRAME[XYPOS(34,54)],GREEN);
661      {NON-SCANNER HEAD STANDBY HEATER }
662      SORC := ONOFF(DIGB,16);

```

Listing of: SOCC.PAS

```
663         PUTOUT(SORC,FRAME[XYPOS(32,12)],GREEN);
664         END;
665     19: BEGIN { NON-SCANNER PED. STANDBY HEATER}
666         SORC := ONOFF(DIGB,12B);
667         PUTOUT(SORC,FRAME[XYPOS(31,12)],GREEN);
668         END;
669     else i := i ;
670     END; { OF CASE }

673 END;

675 { THE FOLLOWING FUNCTIONS ARE USED TO EVALUATE ANALOG
676   DATA FOR BOTH SCANNER AND NON-SCANNER }

679 FUNCTION EQU2(COUNTS: INTEGER): REAL ;
680 BEGIN
681     EQU2 := COUNTS/409.5 ;
682 END;

684 FUNCTION EQU3(COUNTS: INTEGER): REAL;
685 BEGIN
686     EQU3 := -10. +2.*COUNTS/409.5 ;

688 END;

690 FUNCTION EQU4(COUNTS: INTEGER): REAL;
691 BEGIN
692     EQU4 := -187.97 + 37.59*COUNTS/409.5 ;

694 END;

696 FUNCTION EQU5(COUNTS: INTEGER): REAL;
697 BEGIN
698     EQU5 := -2.271*COUNTS/409.5 ;

700 END;

702 FUNCTION EQU6(COUNTS: INTEGER): REAL ;
703 BEGIN
704     EQU6 := -0.8643*COUNTS/409.5 ;

706 END;

708 FUNCTION EQU7(COUNTS: INTEGER): REAL;
709 BEGIN
710     EQU7 := 36. +0.4*COUNTS/409.5 ;

712 END;
```

Listing of: SOCC.PAS

```

714  FUNCTION EQU8(N:REAL):REAL;
715  BEGIN
716      EQU8 := ((((-8.22227099E-19*N+1.04124505E-14)*N-2.6243814E-11)*N
717      +5.04308321E-07)*N+1.44114150E-02)*N-1.21752498E+01 ;

719  END;

721  FUNCTION EQU9(I:INTEGER):REAL;
722  VAR N: REAL;
723  BEGIN
724      N := I/409.5 ;
725      EQU9 := ((( (0.04597458*N-0.4715868)*N+1.605821)*N-1.383922)
726      *N+5.322107)*N-20.0091;

728  END;

731  FUNCTION EQU11EM(I:INTEGER):REAL;
732  BEGIN
733      EQU11EM := ((((-7.8497064E-19*I+1.8092907E-14)*I-1.9129152E-10)
734      *I +1.1525779E-06)*I-9.5425896E-03)*I+5.1936311E+01;

736  END;

738  FUNCTION EQU11FM(I:INTEGER):REAL;
739  BEGIN
740      EQU11FM := ((((-1.0779512E-17*I+2.0178628E-13)*I-1.5103391E-09)
741      *I +5.8237093E-06)*I-2.1023468E-02)*I+6.5527696E+01;

743  END;

745  FUNCTION EQU12EM(I:INTEGER):REAL;
746  BEGIN
747      EQU12EM := ((((-4.9463396E-23*I+1.0318828E-17)*I+8.2795369E-13)
748      *I +9.2748180E-08)*I+1.0514898E-02)*I+2.0765383;

750  END;

752  FUNCTION EQU12FM(I:INTEGER):REAL;
753  BEGIN
754      EQU12FM := ((( (9.5360132E-20*I-1.9623981E-15)*I+1.4936774E-11)
755      *I +7.6635353E-08)*I+1.1301863E-02)*I-3.3441127E+01;

757  END;

759  FUNCTION EQU13(I:INTEGER):REAL;
760  BEGIN
761      EQU13 := I/819.1 ;

763  END;

```

Listing of: SOCC.PAS

```

765  FUNCTION EQU14(N:REAL):REAL;
766  BEGIN
767      EQU14 := ((((-3.7944378*N+45.022096)*N-211.33864)*N+489.0738)
768              *N -583.00496)*N+347.83511;

770  END;

772  FUNCTION EQU15(N:REAL):REAL;
773  BEGIN
774      EQU15 := ((((-2.3382218*N+23.545091)*N-90.571380)*N+168.50204)
775              *N -172.73195)*N+90.962177;

777  END;

779  PROCEDURE ProcessAnalog ;
780  { PROCESS ANALOG DATA FOR BOTH SCANNER AND NONSCANNER }
781  BEGIN
782      CASE MINORFRAMENUM MOD 160 OF
783      { OVERALL PATTERN OCCURS TWICE EACH MAJOR FRAME }
784      { EACH MAJOR FRAME HAS 320 MINOR FRAMES NUMBERED
785      0 TO 319 }

787      125..159: begin { do nothing }
788                  end;

790      4:  BEGIN { SCANNER CHAN 0 ELECTRIC SLICE 2 TEMP NOT AVAILABLE }
791          END;
792      5:  BEGIN { NON-SCANNER CHAN 0 ELECTRIC SLICE 2 TEMP }
793          RANALOG := EQU15(ANALOG/51.0) ;
794          STR(RANALOG:5:3,SORC);
795          LIMITCHECK(RANALOG,-10.0,0.0,40.0,50.0,SORC,ATT);
796          OUTPUT(SORC,FRAME[XYPOS(37,71)],att);
797          END;
798      13: BEGIN { NON-SCANNER CHAN 1 ELECTRIC SLICE 3 TEMP }
799          RANALOG := EQU15(ANALOG/51.0) ;
800          STR(RANALOG:5:3,SORC);
801          LIMITCHECK(RANALOG,-10.,0.0,40.0,50.0,SORC,ATT);
802          OUTPUT(SORC,FRAME[XYPOS(34,71)],GREEN);
803          END;
804      27: BEGIN { SCANNER CHAN 3 POWER CONVERTER TEMP }
805          RANALOG := EQU14(ANALOG/51.0);
806          STR(RANALOG:5:3,SORC);
807          LIMITCHECK(RANALOG,-10.0,0.0,45.0,50.0,SORC,ATT);
808          OUTPUT(SORC,FRAME[XYPOS(9,71)],ATT);
809          END;
810      28: BEGIN { NON-SCANNER CHAN 3 POWER CONVERTER TEMP }
811          RANALOG := EQU14(ANALOG/51.0);
812          STR(RANALOG:5:3,SORC);
813          LIMITCHECK(RANALOG,-10.0,0.0,40.0,50.0,SORC,ATT);
814          OUTPUT(SORC,FRAME[XYPOS(33,71)],ATT);
815          END;

```

Listing of: SOCC.PAS

```

816 35: BEGIN      {SCANNER CHAN 4 BOX BEAM TEMP }
817      RANALOG := EQU14(ANALOG/51.0) ;
818      STR(RANALOG:5:3,SORC);
819      LIMITCHECK(RANALOG,0.0,10.0,34.0,36.0,SORC,ATT);
820      OUTPUT(SORC,FRAME[XYPOS(7,71)],ATT);
821      END;
822 43: BEGIN      { SCANNER CHAN 5   +5 VOLT MONITOR }
823      RANALOG := 2.0*ANALOG/51.0 ;
824      STR(RANALOG:5:3,SORC);
825      LIMITCHECK(RANALOG,3.5,4.0,6.0,6.5,SORC,ATT);
826      OUTPUT(SORC,FRAME[XYPOS(2,71)],ATT);
827      END;
828 45: BEGIN      { NON-SCANNER CHAN 5   +5 VOLT MONITOR }
829      RANALOG := 2.0*ANALOG/51.0 ;
830      STR(RANALOG:5:3,SORC);
831      LIMITCHECK(RANALOG,3.5,4.0,6.0,6.5,SORC,ATT);
832      OUTPUT(SORC,FRAME[XYPOS(39,12)],ATT);
833      END;
834 51: BEGIN      { SCANNER CHAN 6   -15 VOLT MONITOR }
835      RANALOG := -6.0*ANALOG/51.0 ;
836      STR(RANALOG:5:3,SORC);

839 { NEW LIMITS PUT IN 9/29/86... APPLIES TO BOTH NOAA9 & 10 }
840 LIMITCHECK(RANALOG,-16.5,-16.0,-13.5,-13.0,SORC,ATT);
841 { -----*****-----}

844      OUTPUT(SORC,FRAME[XYPOS(5,71)],ATT);
845      END;
846 53: BEGIN      { NON-SCANNER CHAN 6   -15 VOLT MONITOR };
847      RANALOG := -6.0*ANALOG/51.0 ;
848      STR(RANALOG:5:3,SORC);
849      LIMITCHECK(RANALOG,-16.5,-16.0,-14.0,-13.5,SORC,ATT);
850      OUTPUT(SORC,FRAME[XYPOS(39,71)],ATT);
851      END;
852 59: BEGIN      { SCANNER CHAN 7   +15 VOLT MONITOR } ;
853      RANALOG := 6.0*ANALOG/51.0 ;
854      STR(RANALOG:5:3,SORC);
855      LIMITCHECK(RANALOG,13.5,14.0,16.0,16.5,SORC,ATT);
856      OUTPUT(SORC,FRAME[XYPOS(4,71)],ATT);
857      END;
858 60: BEGIN      { NON-SCANNER CHAN 7   +15 VOLT MONITOR}
859      RANALOG := 6.0*ANALOG/51.0;
860      STR(RANALOG:5:3,SORC);
861      LIMITCHECK(RANALOG,13.5,14.0,16.0,16.5,SORC,ATT);
862      OUTPUT(SORC,FRAME[XYPOS(39,52)],ATT);
863      END;
864 67: BEGIN      { SCANNER CHAN 8   +10 VOLT MONITOR }
865      RANALOG := 4.0*ANALOG/51.0 ;
866      STR(RANALOG:5:3,SORC);

```

## Listing of: SOCC.PAS

```

867     LIMITCHECK(RANALOG,8.5,9.0,11.0,11.5,SORC,ATT);
868     OUTPUT(SORC,FRAME[XYPOS(3,71)],ATT);
869     END;
870 68:   BEGIN           { NON-SCANNER CHAN 8 +10 VOLT MONITOR}
871     RANALOG := 4.0*ANALOG/51.0 ;
872     STR(RANALOG:5:3,SORC);
873     LIMITCHECK(RANALOG,8.5,9.0,11.0,11.5,SORC,ATT);
874     OUTPUT(SORC,FRAME[XYPOS(39,32)],ATT);
875     END;
876 99:   BEGIN           { SCANNER CHAN 12 SPACE CRAFT ADAPTER TEMP}
877     RANALOG := EQU15(ANALOG/51.0);
878     STR(RANALOG:5:3,SORC);
879     LIMITCHECK(RANALOG,-10.0,0.0,30.0,40.0,SORC,ATT);
880     OUTPUT(SORC,FRAME[XYPOS(13,71)],ATT);
881     END;
882 100:  BEGIN           {NON-SCANNER SPACE CRAFT ADAPTER TEMP }
883     RANALOG := EQU15(ANALOG/51.0);
884     STR(RANALOG:5:3,SORC);
885     LIMITCHECK(RANALOG,-10.0,0.0,30.0,40.0,SORC,ATT);
886     OUTPUT(SORC,FRAME[XYPOS(35,71)],ATT);
887     END;
888 107:  BEGIN           { SCANNER CHAN 13 PED FOOT TEMP }
889     RANALOG := EQU15(ANALOG/51.0);
890     STR(RANALOG:5:3,SORC);
891     LIMITCHECK(RANALOG,-10.0,0.0,30.0,40.0,SORC,ATT);
892     OUTPUT(SORC,FRAME[XYPOS(14,71)],ATT);
893     END;
894 108:  BEGIN           {NON-SCANNER CHAN 13 PED FOOT TEMP}
895     RANALOG := EQU15(ANALOG/51.0) ;
896     STR(RANALOG:5:3,SORC);
897     LIMITCHECK(RANALOG,-15.0,-5.0,30.0,40.0,SORC,ATT);
898     OUTPUT(SORC,FRAME[XYPOS(36,71)],ATT);
899     END;
900 116:  BEGIN           { SCANNER CHAN 1 ELECTRIC SLICE 3 TEMP }
901     RANALOG := EQU14(ANALOG/51.0) ;
902     STR(RANALOG:5:3,SORC);
903     LIMITCHECK(RANALOG,-10.0,0.0,50.0,55.0,SORC,ATT);
904     OUTPUT(SORC,FRAME[XYPOS(10,71)],ATT);
905     END;
906 117:  BEGIN           {NON-SCANNER CHAN 14 ELEV DRIVE TEMP }
907     RANALOG := EQU14(ANALOG/51.0) ;
908     STR(RANALOG:5:3,SORC);
909     LIMITCHECK(RANALOG,0.0,10.0,30.0,40.0,SORC,ATT);
910     OUTPUT(SORC,FRAME[XYPOS(31,71)],ATT);
911     END;
912 120:  BEGIN           {SCANNER CHAN 15 AZIMUTH DRIVE TEMP }
913     RANALOG := EQU14(ANALOG/51.0);
914     STR(RANALOG:5:3,SORC);
915     LIMITCHECK(RANALOG,-25.0,-15.0,40.0,45.0,SORC,ATT);
916     OUTPUT(SORC,FRAME[XYPOS(8,71)],ATT);
917     END;

```



## Listing of: SOCC.PAS

```

918 124: BEGIN          (NON-SCANNER CHAN 15 AZIMUTH DRIVE TEMP )
919     RANALOG := EQU14(ANALOG/51.0);
920     STR(RANALOG:5:3,SORC);
921     LIMITCHECK(RANALOG,-15.0,-5.0,40.0,50.0,SORC,ATT);
922     OUTPUT(SORC,FRAME[XYPOS(32,71)],ATT);
923     END;
924 else i := i ;
925 END; (OF CASE )
926 END; (OF PROCESSANALOG )

928 FUNCTION SAREF(SAPOS:BYTE):ABC ;
929 BEGIN
930     CASE (SAPOS AND 15) OF          ( CORRECTED 12/11/86 BY WLE )

932 4:     BEGIN
933         SAREF := '0 WINDOW ' ;
934     END;

936 7:     BEGIN
937         SAREF := '0 DEG ' ;
938     END;

940 8:     BEGIN
941         SAREF := '90 WINDOW ' ;
942     END;

944 11:    BEGIN
945         SAREF := '90 DEG ' ;
946     END;

948 12:    BEGIN
949         SAREF := '180 WINDOW';
950     END;

952 15:    BEGIN
953         SAREF := '180 DEG ' ;
954     END;

956 ELSE  SAREF := ' N/A ' ;

958 END; (OF CASE )
959 END; ( OF SAREF FUNCTION )

962 PROCEDURE ProcessScannerDIGA;
963 ( PROCESS DIGITAL 'B' DATA FOR BOTH SCANNER AND NONSCANNER )
964 BEGIN
965     CASE MINORFRAMENUM MOD 40 OF

967 0,12,24,36: BEGIN      ( OUTPUT SCANNER CHANNELS SW LW & TOT)
968                 STR(SSCH:5,SORC);

```

Listing of: SOCC.PAS

```

969          PUTOUT (SORC,FRAME[XYPOS (2,14) ],GREEN);
970          STR (SLCH:5,SORC);
971          PUTOUT (SORC,FRAME[XYPOS (2,34) ],GREEN);
972          STR (STCH:5,SORC);
973          PUTOUT (SORC,FRAME[XYPOS (2,54) ],GREEN);
974          STR (SCPOS:5,SORC);
975          PUTOUT (SORC,FRAME[XYPOS (12,14) ],GREEN);

977          END;
978 37:      BEGIN
979          SSWCSA := EQU2(SDIGA[3]);
980          STR (SSWCSA:5:3,SORC);
981          PUTOUT (SORC,FRAME[XYPOS (9,11) ],GREEN);
982          if (minorframenum<40) then { to account for multiplexing }
983          begin
984              IF ((SCID = $OD) or (scid = $OE)) THEN
985              BEGIN          { DAC LIMITS FOR NOAA-9 }
986                  SSDACV := EQU3(SDIGA[0]);
987                  STR (SSDACV:5:3,SORC);
988                  LIMITCHECK (SSDACV,-0.7,-0.2,0.8,1.3,SORC,ATT);
989                  PUTOUT (SORC,FRAME[XYPOS (3,11) ],ATT);

991                  SLDACV := EQU3(SDIGA[1]);
992                  STR (SLDACV:5:3,SORC);
993                  LIMITCHECK (SLDACV,0.35,0.85,1.85,2.35,SORC,ATT);
994                  PUTOUT (SORC,FRAME[XYPOS (3,31) ],ATT);

996                  STDACV := EQU3(SDIGA[2]);
997                  STR (STDACV:5:3,SORC);
998                  LIMITCHECK (STDACV,-0.47,0.03,1.03,1.53,SORC,ATT);
999                  PUTOUT (SORC,FRAME[XYPOS (3,51) ],ATT);

1001          END
1002          ELSE
1003          BEGIN          { DAC LIMITS FOR NOAA-10 }
1004              SSDACV := EQU3(SDIGA[0]);
1005              STR (SSDACV:5:3,SORC);
1006              LIMITCHECK (SSDACV,1.5,1.5,2.5,2.5,SORC,ATT);
1007              PUTOUT (SORC,FRAME[XYPOS (3,11) ],ATT);

1009              SLDACV := EQU3(SDIGA[1]);
1010              STR (SLDACV:5:3,SORC);
1011              LIMITCHECK (SLDACV,-0.20,-0.20,1.1,1.1,SORC,ATT);
1012              PUTOUT (SORC,FRAME[XYPOS (3,31) ],ATT);

1014              STDACV := EQU3(SDIGA[2]);
1015              STR (STDACV:5:3,SORC);
1016              LIMITCHECK (STDACV,-0.40,-0.40,0.60,0.60,SORC,ATT);
1017              PUTOUT (SORC,FRAME[XYPOS (3,51) ],ATT);
1018          END;

```

Listing of: SOCC.PAS

```

1020      SSMBT := EQU9(SDIGA[5]);
1021      STR(SSMBT:5:3,SORC);
1022      LIMITCHECK(SSMBT,-200.,-20.,35.,200.,SORC,ATT);
1023      PUTOUT(SORC,FRAME[XYPOS(6,11)],ATT);

1025      STMBT := EQU9(SDIGA[7]);
1026      STR(STMBT:5:3,SORC);
1027      LIMITCHECK(STMBT,-25.0,-20.0,35.0,50.0,SORC,ATT);
1028      PUTOUT(SORC,FRAME[XYPOS(6,51)],ATT);

1030      SSMAMT := EQU9(SDIGA[4]);
1031      STR(SSMAMT:5:3,SORC);
1032      LIMITCHECK(SSMAMT,-25.0,-15.0,35.0,45.0,SORC,ATT);
1033      PUTOUT(SORC,FRAME[XYPOS(5,11)],ATT);

1035      STMAMT := EQU9(SDIGA[6]);
1036      STR(STMAMT:5:3,SORC);
1037      LIMITCHECK(STMAMT,-25.0,-15.0,35.0,45.0,SORC,ATT);
1038      PUTOUT(SORC,FRAME[XYPOS(5,51)],ATT);
1039      end;
1040      END;

1042      38:      BEGIN
1043              if (minorframenum <40) then
1044      begin
1045          IF ((SCID = $OD) OR (SCID = $OE)) THEN
1046      BEGIN
1047          SDPBV := EQU4(SDIGA[0]);
1048          STR(SDPBV:5:3,SORC);
1049          LIMITCHECK(SDPBV,-200.0,85.,87.,200.,SORC,ATT);
1050          PUTOUT(SORC,FRAME[XYPOS(10,31)],ATT);
1051          SDNBV := EQU4(SDIGA[1]);
1052          STR(SDNBV:5:3,SORC);
1053          LIMITCHECK(SDNBV,-200.,-87.,-85.,200.,SORC,ATT);
1054          PUTOUT(SORC,FRAME[XYPOS(11,31)],ATT);
1055          STRV1 := EQU5(SDIGA[2]);
1056          STR(STRV1:5:3,SORC);
1057          LIMITCHECK(STRV1,-200.,-6.8,-6.4,200.,SORC,ATT);
1058          PUTOUT(SORC,FRAME[XYPOS(7,31)],ATT);
1059          STRV2 := EQU6(SDIGA[3]);

1061          STR(STRV2:5:3,SORC);
1062          LIMITCHECK(STRV2,-200.,-6.7,-6.3,200.,SORC,ATT);
1063          PUTOUT(SORC,FRAME[XYPOS(8,31)],ATT);

1065      END
1066      ELSE
1067      BEGIN
1068          SDPBV := EQU4(SDIGA[0]);
1069          STR(SDPBV:5:3,SORC);
1070          LIMITCHECK(SDPBV,-200.0,83.0,85.0,200.0,SORC,ATT);

```

Listing of: SOCC.PAS

```

1071      PUTOUT (SORC,FRAME[XYPOS (10,31) ],ATT);
1072      SDNBV := EQU4(SDIGA[1]);
1073      STR(SDNBV:5:3,SORC);
1074      LIMITCHECK(SDNBV,-200.0,-86.5,-84.5,200.0,SORC,ATT);
1075      PUTOUT (SORC,FRAME[XYPOS (11,31) ],ATT);
1076      STRV1 := EQU5(SDIGA[2]);
1077      STR(STRV1:5:3,SORC);
1078      LIMITCHECK(STRV1,-200.0,-7.45,-6.90,200.0,SORC,ATT);
1079      PUTOUT (SORC,FRAME[XYPOS (7,31) ],ATT);
1080      STRV2 := EQU6(SDIGA[3]);

1082      STR(STRV2:5:3,SORC);
1083      LIMITCHECK(STRV2,-200.0,-7.05,-6.50,200.0,SORC,ATT);
1084      PUTOUT (SORC,FRAME[XYPOS (8,31) ],ATT);
1085      END;
1086      SLBBT := EQU8(SDIGA[6]*STRV2/(-6.4));
1087      STR(SLBBT:5:3,SORC);
1088      LIMITCHECK(SLBBT,-25.0,-15.0,45.0,55.0,SORC,ATT);
1089      PUTOUT (SORC,FRAME[XYPOS (5,31) ],ATT);
1090      STBBT := EQU8(SDIGA[7]*STRV2/(-6.4));
1091      STR(STBBT:5:3,SORC);
1092      LIMITCHECK(STBBT,-25.0,-15.0,45.0,55.0,SORC,ATT);
1093      PUTOUT (SORC,FRAME[XYPOS (7,51) ],ATT);
1094      SSWCST := EQU9(SDIGA[5]);
1095      STR(SSWCST:5:3,SORC);
1096      LIMITCHECK(SSWCST,-25.0,-15.0,35.0,45.0,SORC,ATT);
1097      PUTOUT (SORC,FRAME[XYPOS (8,11) ],ATT);
1098      end;
1099      END;
1100      39: BEGIN
1101          if(minorframenum<40) then
1102              begin
1103                  SSDETT := EQU7(SDIGA[1]) ;
1104                  STR(SSDETT:5:3,SORC);
1105                  LIMITCHECK(SSDETT,37.0,37.5,38.5,39.0,SORC,ATT);
1106                  PUTOUT (SORC,FRAME[XYPOS (4,11) ],ATT);
1107              end;
1108          if(minorframenum=79) then
1109              begin
1110                  SLDETT := EQU7(SDIGA[2]);
1111                  STR(SLDETT:5:3,SORC);
1112                  LIMITCHECK(SLDETT,37.0,37.5,38.5,39.0,SORC,ATT);
1113                  PUTOUT (SORC,FRAME[XYPOS (4,31) ],ATT);
1114                  STDETT := EQU7(SDIGA[3]);
1115                  STR(STDETT:5:3,SORC);
1116                  LIMITCHECK(STDETT,37.0,37.5,38.5,39.0,SORC,ATT);
1117                  PUTOUT (SORC,FRAME[XYPOS (4,51) ],ATT);
1118              end;
1119          SCMDE := SDIGA[7] ;
1120          sorc := hexdig[scmde shr 12]
1121                  + hexdig[(scmde shr 8) and 15]

```

## Listing of: SOCC.PAS

```

1122             + hexdig[(scmde shr 4) and 15]
1123             + hexdig[scmde and 15] ;
1124     PUTOUT(SORC,FRAME[XYPOS(14,11)],GREEN);
1125     SSTAT := SDIGA[6] ;
1126     STR(SSTAT:5,SORC);
1127     PUTOUT(SORC,FRAME[XYPOS(16,11)],GREEN);
1128     SAPOSL := SDIGA[5];
1129     SORC := SAREF(SAPOSL);
1130     SAPOSH := SDIGA[4] ;
1131     PUTOUT(SORC,FRAME[XYPOS(14,31)],GREEN);
1132     SAPOS := ((SAPOSH AND 255) SHL 4 )
1133             OR ((SAPOSL AND 240) SHR 4);
1134     STR(SAPOS:6,SORC);
1135     PUTOUT(SORC,FRAME[XYPOS(15,31)],GREEN);
1136     SAZP := SAPOS * 0.075 ;
1137     STR(SAZP:6:3,SORC);
1138     PUTOUT(SORC,FRAME[XYPOS(16,31)],GREEN);
1139     END;
1140 else i := i ; { do nothing }
1141     END; { OF CASE }
1142 END; { OF PROCEDURE PROCESSSCANNERDIGA }

1145 FUNCTION NEPOS(NDIGA:INTEGER): ABC ;
1146     { NONSCANNER ELEVATION POSITION}

1148 BEGIN
1149     CASE (NDIGA AND 15) OF

1151 1: BEGIN
1152     NEPOS := '180 DEG' ;
1153     END;

1155 9: BEGIN
1156     NEPOS := '180 WINDOW' ;
1157     END;

1159 3: BEGIN
1160     NEPOS := '78 DEG' ;
1161     END;

1163 11: BEGIN
1164     NEPOS := '78 WINDOW' ;
1165     END;

1167 5: BEGIN
1168     NEPOS := '0 DEG' ;
1169     END;

1171 13: BEGIN
1172     NEPOS := '0 WINDOW' ;

```

Listing of: SOCC.PAS

```

1173         END;

1175     ELSE NEPOS := 'UNDEFINED ' ;
1176     END; { OF CASE }
1177     END; { OF NEPOS FUNCTION }

1180     FUNCTION NAREF (NDIGA: INTEGER): ABC; { NONSCANNER AZIMUTH POSITION }
1181     BEGIN
1182         CASE (NDIGA AND 15) OF

1184     4:     BEGIN
1185             NAREF := '0 WINDOW ' ;
1186             END;
1187     7:     BEGIN
1188             NAREF := '0 DEG ' ;
1189             END;

1191     8:     BEGIN
1192             NAREF := '90 WINDOW ' ;
1193             END;

1195     11:    BEGIN
1196             NAREF := '90 DEG ' ;
1197             END;

1199     12:    BEGIN
1200             NAREF := '180 WINDOW ' ;
1201             END;

1203     15:    BEGIN
1204             NAREF := '180 DEG ' ;
1205             END;

1207     ELSE
1208         NAREF := 'UNDEFINED ' ;

1210     END; { OF CASE }
1211     END; { OF NAREF FUNCTION }

1214     PROCEDURE ProcessNonScannerDIGA ; { PROCESS NONSCANNER DIGITAL 'A' DATA }
1215     VAR POSIT, NS: REAL; NPOSIT : INTEGER ;
1216     BEGIN
1217         CASE MinorFrameNum MOD 160 OF
1218             { OVERALL PROCESS OCCURS TWICE EVERY MAJOR FRAME }

1220     3..7,
1221     11..15, 19..23, 27..31, 35..39, 43..47, 51..55, 59..63, 67..71,
1222     75..79, 83..87, 91..95, 99..103, 107..111, 115..119, 123..127, 131..135,
1223     139..143, 147..151, 155..159:

```

Listing of: SOCC.PAS

```

1225     BEGIN
1226     CASE MINORFRAMENUM MOD 8 OF
1227     3: BEGIN
1228         STR(NDIGA:5,SORC);      {NMFTCH}
1229         PUTOUT(SORC,FRAME[XYPOS(22,12)],GREEN);
1230     END;
1231     4: BEGIN
1232         STR(NDIGA:5,SORC);      {NMSCH}
1233         PUTOUT(SORC,FRAME[XYPOS(22,32)],GREEN);
1234     END;
1235     5: BEGIN
1236         STR(NDIGA:5,SORC);      {NWTCH}
1237         PUTOUT(SORC,FRAME[XYPOS(22,52)],GREEN);
1238     END;
1239     6: BEGIN
1240         STR(NDIGA:5,SORC);      {NWSCH}
1241         PUTOUT(SORC,FRAME[XYPOS(22,71)],GREEN);
1242     END;
1243     7: BEGIN
1244         STR(NDIGA:5,SORC);      {NSMCH}
1245         PUTOUT(SORC,FRAME[XYPOS(29,12)],GREEN);
1246     END;
1247     END; { OF INNER CASE }
1248 END;
1249 0: BEGIN
1250     SORC := HEXDIG[NDIGA SHR 12]
1251         + HEXDIG[(NDIGA SHR 8) AND 15]
1252         + HEXDIG[(NDIGA SHR 4) AND 15]
1253         + HEXDIG[NDIGA AND 15] ;
1254     { STR(NDIGA:5,SORC); }      { COMMAND ECHO }
1255     PUTOUT(SORC,FRAME[XYPOS(36,52)],GREEN);
1256 END;
1257 1: BEGIN
1258     STR(NDIGA:5,SORC);          { INSTRUMENT STATUS }
1259     PUTOUT(SORC,FRAME[XYPOS(37,52)],GREEN);
1260 END;
1261 2: BEGIN
1262     SORC := NEPOS(NDIGA);      { NEPOS }
1263     PUTOUT(SORC,FRAME[XYPOS(32,30)],GREEN);
1264 END;
1265 10,90: BEGIN
1266     SORC := NAREF(NDIGA) ;      { NAREF }
1267     PUTOUT(SORC,FRAME[XYPOS(34,30)],GREEN);
1268     NPOSIT := NDIGA SHR 4 ;
1269     STR(NPOSIT:5,SORC);
1270     PUTOUT(SORC,FRAME[XYPOS(35,30)],GREEN);      { NAPOS }
1271     POSIT:= NPOSIT * 0.075 ;
1272     STR(POSIT:10:3,SORC);      { NADEG }
1273     PUTOUT(SORC,FRAME[XYPOS(36,30)],GREEN);
1274 END;

```

## Listing of: SOCC.PAS

```

1275 18,98:  BEGIN
1276          NS := EQU12FM(NDIGA) ;
1277          STR(NS:5:2,SORC);  { NMFTHT }
1278          LIMITCHECK(NS,10.0,20.0,40.0,50.0,SORC,ATT);
1279          PUTOUT(SORC,FRAME[XYPOS(25,12)],ATT);
1280          END;

1282 24,104:  BEGIN
1283          NS := EQU13(NDIGA);
1284          STR(NS:6:4,SORC);      { NSWCSA }

          PUTOUT(SORC,FRAME[XYPOS(27,71)],GREEN);
1286          END;
1287 25,105:  BEGIN
1288          NS := EQU11FM(NDIGA);
1289          STR(NS:5:2,SORC);      { NSWCSA }
1290          LIMITCHECK(NS,0.0,10.0,30.0,40.0,SORC,ATT);
1291          PUTOUT(SORC,FRAME[XYPOS(26,71)],ATT);
1292          END;
1293 26,106:  BEGIN
1294          NS := EQU12FM(NDIGA);
1295          STR(NS:5:2,SORC);      { NMFSHT }
1296          LIMITCHECK(NS,10.0,20.0,40.0,50.0,SORC,ATT);
1297          PUTOUT(SORC,FRAME[XYPOS(25,32)],ATT);
1298          END;
1299

1301 33,113:  BEGIN
1302          NS := EQU13(NDIGA);
1303          STR(NS:6:4,SORC);      { NTREFV }
1304          LIMITCHECK(NS,4.7,4.9,5.1,5.3,SORC,ATT);
1305          PUTOUT(SORC,FRAME[XYPOS(38,32)],ATT);
1306          END;

1308 34,114:  BEGIN
1309          NS := EQU12FM(NDIGA);
1310          STR(NS:5:2,SORC);      { NWFTHT }
1311          LIMITCHECK(NS,10.0,20.0,40.0,50.0,SORC,ATT);
1312          PUTOUT(SORC,FRAME[XYPOS(25,52)],ATT);
1313          END;

1315 40:      BEGIN
1316          NS := EQU11FM(NDIGA);
1317          STR(NS:5:2,SORC);      { NMFTAT }
1318          LIMITCHECK(NS,10.0,20.0,40.0,50.0,SORC,ATT);
1319          PUTOUT(SORC,FRAME[XYPOS(24,12)],ATT);
1320          END;

1322 41:      BEGIN
1323          NS := EQU11FM(NDIGA);
1324          STR(NS:5:2,SORC);      { NMFTLT }
1325          LIMITCHECK(NS,-10.0,0.0,40.0,50.0,SORC,ATT);

```



## Listing of: SOCC.PAS

```

1326          PUTOUT (SORC,FRAME[XYPOS (23,12) ],ATT);
1327          END;

1329  42,122:  BEGIN
1330          NS := EQU12FM(NDIGA);
1331          STR(NS:5:2,SORC);      { NWF SHT }
1332          LIMITCHECK(NS,10.0,20.0,40.0,50.0,SORC,ATT);
1333          PUTOUT (SORC,FRAME[XYPOS (25,71) ],ATT);
1334          END;

1336  50,130:  BEGIN
1337          NS := EQU12FM(NDIGA);
1338          STR(NS:5:2,SORC);      { NWFBBT }
1339          LIMITCHECK(NS,0.0,10.0,55.0,60.0,SORC,ATT);
1340          PUTOUT (SORC,FRAME[XYPOS (27,52) ],ATT);
1341          END;

1343  58,138:  BEGIN
1344          NS := EQU12FM(NDIGA);
1345          STR(NS:5:2,SORC);      { NMF SPT }
1346          LIMITCHECK(NS,0.0,10.0,30.0,40.0,SORC,ATT);
1347          PUTOUT (SORC,FRAME[XYPOS (26,12) ],ATT);
1348          END;

1350  64:      BEGIN
1351          NS := EQU11FM(NDIGA);
1352          STR(NS:5:2,SORC);      { NMF SAT }
1353          LIMITCHECK(NS,10.0,20.0,40.0,50.0,SORC,ATT);
1354          PUTOUT (SORC,FRAME[XYPOS (24,32) ],ATT);
1355          END;

1357  65:      BEGIN
1358          NS := EQU11FM(NDIGA);
1359          STR(NS:5:2,SORC);      { NMF SLT }
1360          LIMITCHECK(NS,-10.0,0.0,40.0,50.0,SORC,ATT);
1361          PUTOUT (SORC,FRAME[XYPOS (23,32) ],ATT);
1362          END;

1364  66,146:  BEGIN
1365          NS := EQU12FM(NDIGA);
1366          STR(NS:5:2,SORC);      { NMFBBT }
1367          LIMITCHECK(NS,0.0,10.0,55.0,60.0,SORC,ATT);
1368          PUTOUT (SORC,FRAME[XYPOS (27,12) ],ATT);
1369          END;
1370  73:      BEGIN
1371          NS := EQU13(NDIGA);
1372          STR(NS:6:4,SORC);      { NCHTRV }
1373          PUTOUT (SORC,FRAME[XYPOS (27,32) ],GREEN);
1374          END;

1376  74,154:  BEGIN

```

## Listing of: SOCC.PAS

```

1377          NS := EQU12FM(NDIGA);
1378          STR(NS:5:2,SORC);          {NWFSPT }
1379          LIMITCHECK(NS,0.0,10.0,30.0,40.0,SORC,ATT);
1380          PUTOUT(SORC,FRAME[XYPOS(26,52)],ATT);
1381          END;

1383 80:      BEGIN
1384          NS := EQU11FM(NDIGA);
1385          STR(NS:5:2,SORC);          { NSMHT }
1386          LIMITCHECK(NS,-10.0,0.0,40.0,50.0,SORC,ATT);
1387          PUTOUT(SORC,FRAME[XYPOS(29,32)],ATT);
1388          END;

1390 81:      BEGIN
1391          NS := EQU11FM(NDIGA);
1392          STR(NS:5:2,SORC);          { NSMAT }
1393          LIMITCHECK(NS,-10.0,0.0,40.0,50.0,SORC,ATT);
1394          PUTOUT(SORC,FRAME[XYPOS(29,52)],ATT);
1395          END;

1397 82:      BEGIN
1398          NS := EQU11FM(NDIGA);
1399          STR(NS:5:2,SORC);          { NSMBT }
1400          LIMITCHECK(NS,-10.0,0.0,40.0,50.0,SORC,ATT);
1401          PUTOUT(SORC,FRAME[XYPOS(29,71)],ATT);
1402          END;

1404 120:     BEGIN
1405          NS := EQU11FM(NDIGA);
1406          STR(NS:5:2,SORC);          {NWFTAT }
1407          LIMITCHECK(NS,10.0,20.0,40.0,50.0,SORC,ATT);
1408          PUTOUT(SORC,FRAME[XYPOS(24,52)],ATT);
1409          END;

1411 121:     BEGIN
1412          NS := EQU11FM(NDIGA);
1413          STR(NS:5:2,SORC);          {NWFTLT}
1414          LIMITCHECK(NS,-10.0,0.0,40.0,50.0,SORC,ATT);
1415          PUTOUT(SORC,FRAME[XYPOS(23,52)],ATT);
1416          END;

1418 144:     BEGIN
1419          NS := EQU11FM(NDIGA);
1420          STR(NS:5:2,SORC);          { NWFSAT }
1421          LIMITCHECK(NS,10.0,20.0,40.0,50.0,SORC,ATT);
1422          PUTOUT(SORC,FRAME[XYPOS(24,71)],ATT);
1423          END;

1425 145:     BEGIN
1426          NS := EQU11FM(NDIGA);
1427          STR(NS:5:2,SORC);          {NWFSLT }

```

## Listing of: SOCC.PAS

```

1428          LIMITCHECK(NS,-10.0,0.0,40.0,50.0,SORC,ATT);
1429          PUTOUT(SORC,FRAME[XYPOS(23,71)],ATT);
1430          END;

1432  153:  BEGIN
1433          NS := EQU11FM(NDIGA);
1434          STR(NS:5:2,SORC); { NBEET }
1435          LIMITCHECK(NS,-10.0,0.0,40.0,50.0,SORC,ATT);
1436          PUTOUT(SORC,FRAME[XYPOS(26,32)],ATT);
1437          END;
1438  else i := i ; { do nothing }
1439  END; { OF CASE }
1440  END; { OF ProcessNonScannerDIGA }

1442  procedure stat; { STATUS OF ADCCP COMMUNICATIONS }
1443  begin
1444          case status of

1446  0:  begin
1447          sorc := '
1448          att := green;
1449          end;

1451  1:  begin
1452          sorc := 'protocol error ' ;
1453          att := red ;
1454          end;

1456  2:  begin
1457          sorc := 'checksum error ' ;
1458          att := red ;
1459          end;

1461  3:  begin
1462          sorc := 'timeout error ' ;
1463          att := red ;
1464          end;
1465  else i := i ; { do nothing }
1466  end; { of case }
1467          putout(sorc,frame[xypos(44,2)],att);

1469  end;

1471  procedure SCRDDUMP(var i,j: integer) ;
1472  { 80 COLUMN BY 50 ROW SCREEN DUMP TO PRINTER }
1473  TYPE CHARBUFF = ARRAY[0..8000] OF CHAR ;
1474  VAR CFRAME: CHARBUFF ABSOLUTE #B800:#0000;
1475  PRFRAME: ARRAY[0..4000] OF CHAR ;
1476  K ,l: INTEGER ;
1477  begin
1478  IF (I+J = 0) THEN

```

Listing of: SOCC.PAS

```

1479 BEGIN
1480 FOR K := 0 TO 3939 DO
1481 BEGIN
1482 PRFRAME[K] := CFRAME[K*2];
1483 END;
1484 END;
1485 for l:= 0 to 4 do
1486 begin
1487     if (j<79) then
1488         WRITE(LST,PRFRAME[I*80 +j]) else writeln(lst,PRFRAME[I*80+j]);
1489     j:= j+1;
1490 end;
1491 if (j>79) then
1492 begin
1493     j:=0;
1494     i := i+1;
1495     if (i>45) then
1496     begin
1497         i := 0;
1498         statpr :=0;
1499     end;
1500 end;
1501 end;

1503 procedure dumpbuff;
1504 { THIS ROUTINE IS CALLED WHEN A CONTROL D IS HIT. IT WAS USED FOR
1505 DEBUGGING PURPOSES }
1506 var line : string[80] ; j,k,l: integer ; bit: byte ;
1507 begin

1509     for j := 0 to 4 do
1510     begin
1511         K := 0 ;
1512         for i := 0 to 5 do
1513         begin
1514             line := ' ' ;
1515             for l:= 0 to 19 do
1516             begin
1517                 bit := minorframe[j,k];
1518                 line := line + hexdig[bit shr 4] + hexdig[bit and $f] + ' ' ;
1519                 k := k + 1 ;
1520             end;
1521             writeln(lst,line);
1522         end;
1523         writeln (lst );
1524     end;
1525     statpr := 1 ;
1526 end;

1528 PROCEDURE CLEARBUF ; { CLEAR THE MINORFRAME BUFFER }
1529 VAR I,J:INTEGER ;

```

Listing of: SOCC.PAS

```

1530  BEGIN
1531      FOR I := 0 TO 4 DO
1532          BEGIN
1533              FOR J:= 0 TO 103 DO
1534                  MINORFRAME[I,J] := 0 ;
1535          END;
1536  END;

1539  procedure eoffile(segm,offs:integer);
1540  var i : integer ;
1541  { this procedure positions a file to the eof. Turbo Pascal would not
1542  do this in the case of files with more than 32767 records. This
1543  routine simply moves the number of records contained in bytes 4
1544  through 7 in the file control block to the current record number
1545  bytes 44 through 47. }
1546      begin
1547          for i:= 4 to 7 do
1548              begin
1549                  memw[segm:offs+i+40] := mem[segm:offs+i] ;
1550              end;
1551      end;

1553  { BEGINNING OF THE MAIN PROGRAM-----}

1557  BEGIN
1558      SCID := 0 ; { INITIALIZE SPACE CRAFT ID'S }
1559      OLDSCID := 0 ;{ THESE TWO VARIABLES ARE USED TO DETECT A
1560      CHANGE IN SATELLITE ID, THUS ENABLING A CLEARING OF THE
1561      SCREEN FOR THE NEW SATELLITE BEING DISPLAYED }
1562      ESIFL := OFF ; { INITIALIZE SCANNER PULSE LOAD TO OFF }
1563      ASSIGN (F9,NOAA9);
1564      RESET(F9);
1565      eoffile(seg(F9),ofs(F9));
1566      ASSIGN(F10,NOAA10);
1567      RESET(F10);
1568      EOFFILE(SEG(F10),OFS(F10));
1569      STORINT[0] := SCRINT[0] ; { SAVE PRINT SCREEN VECTOR }
1570      STORINT[1] := SCRINT[1] ;
1571      SCRINT[0] := OFS(PRSTAT) ; { PLACE NEW VECTOR TO OUR ROUTINE }
1572      SCRINT[1] := CSEG ;
1573      NEWSCREEN ; { SET UP 80 X 50 DISPLAY SCREEN }
1574      DISPLAYACRO ; { DISPLAY ACRONYMS ON SCREEN }
1575      STATPR := 0; linenum := 0; charnum:= 0;
1576      repeat
1577          INITRCV ;
1578          STAT;
1579          key := GETKEY ;
1580          if (key = 24 ) then status := 0 ;

```

Listing of: SOCC.PAS

```

1581      until status = 0 ;                { WAITING FOR ERROR = 0 }
1582      MINORFRAMENUM := -1 ;

1584 REEP:  GETBUF ;
1585        STAT;
1586        key := GETKEY ;
1587        if (key = 2 ) then goto stop ; { CONTROL B }
1588        if (key = 18 ) then displayacro ; { CONTROL R }
1589        if (key = 4 ) then dumpbuff ; { CONTROL D }
1590        IF (STATUS <> 0 ) THEN GOTO REEP ;
1591        { IF WE HAVE A BAD STATUS, THEN TRY AGAIN }
1592        IF (LENG = 0 ) THEN GOTO REEP ;
1593        { IF WE HAVE NO DATA, TRY AGAIN }
1594        FOR MN := 0 TO 4 DO
1595          { WE HAVE GOOD DATA, SO PROCESS 5 MINOR FRAMES }
1596          BEGIN
1597            DIVY(MN) ; { EXTRACT USEFUL DATA FROM TIP MINOR FRAME MN }
1598            if (minorframenum >319) then goto bottom ;
1599            IF ((FORG<>' 9') AND (FORG <> '10')) THEN GOTO BOTTOM ;
1600            ProcessDigb ;
1601            ProcessAnalog ;
1602            ProcessScannerDiga;
1603            ProcessNonscannerDiga;
1604            str(minorframenum:5,sorc); {DISPLAY MINOR FRAME NUMBER }
1605            putout(sorc,frame[x ypos(42,15)],white);
1606 bottom:  if statpr = 1 then scrdump(linenum,charnum);
1607          END;
1608          goto reep ;
1609 STOP:  SCRINT[0] := STORINT[0] ;
1610        {RESTORE NORMAL PRINT SCREEN VECTOR }
1611        SCRINT[1] := STORINT[1] ; { DITTO }

1613      CLOSE(F9); { CLOSE NOAA-9 FILE }
1614      CLOSE(F10); { CLOSE NOAA-10 FILE }
1615      OLDScreen ; { CHANGE SCREEN MODE BACK TO NORMAL }
1616      END. { of socc disp program }

```

## APPENDIX B - SCANCHK

SCANCHK is designed to examine scan position data for NOAA-9 and NOAA-10 data acquired by the SOCC monitoring program.

## Listing of: SCANCHK.PAS

```

1  PROGRAM SCANCHK;
2  (
3      SCANCHK is designed to examine scan position data for NOAA-9 and
4      NOAA-10 data acquired by the SOCC monitoring program.

6  written by:    William L. Edmonds
7                  STX Corp.

10 )
11 LABEL REEP ,stop,bottom;
12 TYPE ABC = STRING[80] ;
13 TYPE BYTEBUFF = ARRAY[0..8000] OF BYTE ;
14 TYPE integBUFF = ARRAY[0..4000] OF INTEGER ;
15 TYPE BITE = ARRAY[0..80] OF BYTE ;
16 TYPE MinorFrameBufs = Array[0..103] of byte ;
17 Type dualbufs = Array[0..4] of MinorFrameBufs ;
18 TYPE MinorFrameWords = Array[0..51] OF INTEGER ;
19 Type dualwordbufs = Array[0..4] of MinorFrameWords ;
20 TYPE ERBREC = ARRAY[0..21] OF BYTE ; ( USED IN SAVING ERBE DATA TO DISK
21 TYPE REGPACK = RECORD
22     ax,bx,cx,dx,bp,di,si,ds,es,flags:integer;
23     END;
24 TYPE COMBUFF = ARRAY[0..519] OF BYTE ;
25 type smallbufw = array[0..3] of integer ;
26 TYPE COMBUFFW = ARRAY[0..259] OF INTEGER ;
27 CONST
28     HEXDIG : ARRAY[0..15] OF CHAR = '0123456789ABCDEF' ;
29     ON : STRING[3] = 'ON' ;
30     OFF : STRING[3] = 'OFF' ;

32     WHITE: INTEGER = 15 ;
33     RED: INTEGER = 12 ;
34     YELLOW : INTEGER = 14 ;
35     GREEN : INTEGER = 10 ;
36     msgrcv : byte = 2 ;
37     xmtrcv : byte = 7 ;
38     maxsec : integer = 10 ;
39     sync : byte = #32 ;
40     bisync : integer = #220 ;
41     noerr : integer = 0 ;
42     ptlerr : integer = 1 ;
43     cserr : integer = 2 ;
44     timerr : integer = 3 ;
45     id0msg : array[0..3] of byte = (#32,#32,0,#8) ;
46     id1msg : array[0..3] of byte = (#32,#32,0,#c) ;
47     idmsln : integer = 4 ;
48 VAR
49     spacecount : integer ;
50     Spacelook : integer;

```



## Listing of: SCANCHK.PAS

```

51     Sample9,sample10 : integer ;
52     Sample71 : integer ;
53     Sample74 : integer ;
54     days,hrs,mins :integer ;    millisecs,secs :real ;
55     ESIFL : ABC ;
56     ERBDAT : ERBREC ;    ( THIS ARRAY HOLDS ERBE DATA TO BE SAVED TO DISK )
57     FL : FILE OF ERBREC ;
58     key : integer ;
59     state : byte ;
60     SCID : BYTE ;    ( SCID SHOULD BE = HEX "D" )
61     id : byte ;
62     IDNUM : INTEGER ;
63     msgid : byte ;
64     elpmin : byte ;
65     elpsec : byte ;

67     RECPACK : REGPACK ;
68     ah,al,ch,cl,dh : byte ;
69     TIME : ARRAY[0..5] OF BYTE ;
70     MN : INTEGER ;
71     status,count: integer ;
72     linenum,charnum: integer ;
73     MinorFrameNUM : integer ;
74     MajorFrameNUM : integer ;
75     Digb : byte;
76     scrnmode: array[0..15] of byte ;
77     SDIGA : ARRAY[0..7] OF INTEGER ;
78     NDIGA: INTEGER ;
79     ANALOG : byte ;
80     analogint : integer ;
81     RANALOG : REAL ;

83     VALST : ABC ;
84     LENG : INTEGER ;
85     RCVBUF : COMBUFF ABSOLUTE #8F80: #0000 ;
86     RCVBUFW : COMBUFW ABSOLUTE #8F80:#0004 ;
87     msgbuf : smallbufw absolute #8F80: #0000 ;
88     XMTBUF : COMBUFF ABSOLUTE #8F80: #0400 ;
89     MinorFrame : dualbuffs absolute #7f80 : #0004 ;    ( 9800:0004 ; )
90     MinorFrameW : dualwordbuffs absolute #7f80 : #0004 ;    ( 9800:0004 ; )
91     MINFW : COMBUFW ABSOLUTE #7f80 : #0004 ;    ( 9800:0004 ; )
92     FRAME: integBUFF ABSOLUTE #B800:#0000;
93     BFRAME: BYTEBUFF ABSOLUTE #B800:#0000;
94     txt : text ;
95     txtfile : string[10] ;
96     SORC:ABC ;
97     ATT, XY: INTEGER;    I,K,    J:INTEGER;
98     SCRINT: ARRAY[0..1] OF INTEGER ABSOLUTE #0000:#0014;
99     STORINT : ARRAY[0..1] OF INTEGER ;
100    statpr : byte absolute #0050:#0000;

```

Listing of: SCANCHK.PAS

```

105  {-----scanner digital "a" variables-----}
106  TYPE COUNTS = INTEGER;
107  VAR
108  SSCH,SLCH,STCH : COUNTS ;    { CHANNEL OUTPUT }
109  SSWCSA,SSWCST : REAL ;      { SWICS AMP OUTPUT & TEMP }
110  SSDACV,SLDACV,STDACV : REAL ; { DAC VOLTAGES }
111  SDPBV,SDNBV : REAL ;       { POS & NEG DETECTOR BIAS VOLTAGES}
112  STRV1,STRV2 : REAL ;      { TEMP REF VOLTAGES}
113  SSDETT,SLDETT,STDETT : REAL ; { DETECTOR TEMPS}
114  SLBBT,STBBT : REAL ;     { BLACKBODY TEMPS }
115  SSMBT,STMBT : REAL ;     { MAM BAFFLE TEMPS }
116  SSMAMT,STMAMT : REAL ;   { MAM TEMPS }
117  SCMDE,SSTAT,SCPOS : COUNTS ; { COMMAND ECHO, STATUS, SCAN POSITION}
118  SAPOSL,SAPOSH: BYTE ;   { LOW & HIGH AZIMUTH POSITION BYTES }
119  SAZP : REAL ; { AZIMUTH POSITION }
120  SAPOS : COUNTS ;
121  TREF : REAL ; { TEMP REAL NUM }

123  { -----EXTERNALS-----}

127  PROCEDURE S5080(var i :byte); EXTERNAL 'CONO.COM';
128  PROCEDURE PUTOUT(VAR SORC:ABC;VAR FRAME:INTEGER;ATTR:INTEGER);
129  EXTERNAL 'PUTOUT.COM';
130  FUNCTION FRSTAT:INTEGER; EXTERNAL 'FRSTAT.COM';

133  { -----PROCEDURES & FUNCTIONS-----}

135  PROCEDURE OUTPUT(VAR SORC: ABC ; VAR FRAME : INTEGER ; ATTR : INTEGER );
136  VAR BLANKS : ABC ;
137  BEGIN
138  { BLANKS := ' ' ; 11 BLANKS }
139  { PUTOUT(BLANKS, FRAME , WHITE ); }
140  PUTOUT(SORC,FRAME,ATTR);
141  END;

143  function cntlb : integer ;
144  begin
145  with reckpt do
146  begin
147  ah := 6 ;
148  al := 0;
149  ax := ah shl 8 + al ;
150  dx := $ff ;
151  end;
152  intr($21,reckpt);

```

## Listing of: SCANCHK.PAS

```
153         with recpack do
154         begin
155             al := ax and $ff ;
156         end;
157         cntlb := al ;
158     end;

163     FUNCTION XYPOS(ROW,COL: INTEGER ): INTEGER ;
164     BEGIN
165         XYPOS := ROW * 80 + COL;
166     END;

168     FUNCTION ONOFF(DIGB,I:BYTE):ABC ;
169     BEGIN
170         ONOFF := ON ;
171         IF((DIGB AND I)>0 ) THEN ONOFF := OFF ;
172     END;

176     procedure NEWSSCREEN ;
177     BEGIN
178         SCRNMODE[0] := $71;
179         SCRNMODE[1] := $50;
180         SCRNMODE[2] := $5A;
181         SCRNMODE[3] := $0F;
182         SCRNMODE[4] := $1B;
183         SCRNMODE[5] := 6;
184         SCRNMODE[6] := $19;
185         SCRNMODE[7] := $1A;
186         SCRNMODE[8] := 3;
187         SCRNMODE[9] := 7;
188         SCRNMODE[10] := $20 ;
189         SCRNMODE[11] := $20 ;
190         SCRNMODE[12] := 0;
191         SCRNMODE[13] := 0;
192         SCRNMODE[14] := 0;
193         SCRNMODE[15] := 0;
194         S5080(SCRNMODE[0]);
195         PORT[$3D8] := $19 ;
196         PORT[$3E0] := $18 ;
197         PORT[$3D9] := 0 ;

200     END;

202     PROCEDURE OLDSCREEN ;
203     VAR LOC : INTEGER ;
```

## Listing of: SCANCHK.PAS

```

204 BEGIN
205     FOR LOC := 0 TO 3999 DO
206         FRAME[LOC] := #F00 ;

208         SCRNMODE[4] := #1F ;
209         SCRNMODE[7] := #1C ;
210         SCRNMODE[8] := 2;
211         SCRNMODE[10] := 6;
212         SCRNMODE[11] := 7;
213         S5080(SCRNMODE[0]);
214     END;

219 PROCEDURE DisplayTime;
220 VAR TIMSTRING : ABC ;
221 BEGIN
222     days := (time[0]shr 1) + ((time[1] and 128)shr 7) ;
223     millisecs := (((time[1]and 7)*256.0 + time[2])*256.0 + time[3])*256.0
224     hrs := trunc(millisecs/3600000.0) ;
225     mins := trunc(millisecs/60000.0) mod 60 ;
226     secs := trunc((millisecs/1000.0)-mins*60.0-hrs*3600.0);
227     str(days:4,sorc); putout(sorc,frame[xypos(42,40)],white);
228     str(hrs:2,sorc); putout(sorc,frame[xypos(42,45)],white);
229     str(mins:2,sorc); putout(sorc,frame[xypos(42,50)],white);
230     str(secs:6:3,sorc); putout(sorc,frame[xypos(42,55)], white);
231 END;

235 PROCEDURE DIVY ;      { reads TIP MINOR FRAME data from disk}
236 LABEL RETURN ;
237 var nextmf,tipstatus: integer ;   nmf,mf: string[4] ;
238 BEGIN
239     READ(FL,ERBDAT);
240     nextmf := minorframenum +1 ;
241     if (nextmf > 319 ) then nextmf := 0 ;
242     MINORFRAMENUM := ERBDAT[1] + (ERBDAT[0] AND 1)shr 8;
243     if (minorframenum<>nextmf) then
244     begin
245         str(nextmf:4,nmf);
246         str(minorframenum:4,mf);
247         sorc := 'expecting mf ' + nmf + ' but found mf ' + mf ;
248         putout(sorc,frame[xypos(45,2)],yellow);
249     end ;
250     if (minorframenum = 0 ) then
251     begin
252         sorc := '
253         putout(sorc,frame[xypos(45,2)],yellow);
254     FOR I := 0 TO 4 DO

```

Listing of: SCANCHK.PAS

```

255         TIME[I] := ERBDAT[I+2] ;
256         displaytime;
257     END;
258     DIGB := ERBDAT[6];
259     ANALOG := ERBDAT[7] ;
260     analogint := analog ;
261     NDIGA := (ERBDAT[14] shl 8) or ERBDAT[15] ;
262     SDIGA[0] := (ERBDAT[8] shl 8 or ERBDAT[9] )shr 4 ;
263     SDIGA[1] := (((ERBDAT[9] AND 15) SHL 8) OR ERBDAT[10]);
264     SDIGA[2] := (ERBDAT[11] SHL 4) OR ( ERBDAT[12] SHR 4) ;
265     SDIGA[3] := ((ERBDAT[12] and 15)shl 8 or ERBDAT[13] ) ;
266     SDIGA[4] := (ERBDAT[16] shl 8 or ERBDAT[17] )shr 4;
267     SDIGA[5] := (((ERBDAT[17] AND 15 ) SHL 8) OR ERBDAT[18]);
268     SDIGA[6] := (ERBDAT[19] SHL 4) OR (ERBDAT[20] SHR 4 ) ;
269     SDIGA[7] :=((( ERBDAT[20] and 15) shl 8 or ERBDAT[21] ) ;
270     ssch := sdiga[0] ;
271     slch := sdiga[1] ;
272     stch := sdiga[2] ;
273     scpos := sdiga[3] ;
274     RETURN;
275     END; { OF DIVY }

278     Procedure ScanPos ;

280     Begin
281         Case Minorframenum mod 40 of

283     0,1,2 : begin
284     {
285         Calculate sum of space look data.

287     }

288         Spacelook := Spacelook + Sdiga[3] + Sdiga[7] ;
289         spacecount := spacecount + 2 ;
290         end;
291     3 : begin
292     {
293         Complete the sum of space look data and calculate average.

295     }

296         Spacelook := Spacelook + Sdiga[3] + Sdiga[7];
297         spacecount := spacecount + 2 ;
298         Spacelook := Spacelook div 8 ;
299         sorc := 'SPACE ' ; output(sorc,frame[xypos(17,2)],white);
300         str(spacelook:5,sorc) ; output(sorc,frame[xypos(17,1)],green);
301         end;

303     4 : begin
304     {
305         Get first and second Earth scan positions.

```

Listing of: SCANCHK.PAS

```

307   }
308       Sample9 := Sdiga[3] ; sample10 := sdiga[7] ;
309       sorc := 'Sample 9 ' ; output(sorc,frame[xypos(17,21)],white);
310       str(sample9:5,sorc) ; output(sorc,frame[xypos(17,31)],green);
311   end;

313 35:   begin
314   {
315       Get first internal cal position.

317   }
318       Sample71 := Sdiga[3] ;
319       sorc := 'Sample 71 ' ; output(sorc,frame[xypos(17,41)],white);
320       str(sample71:5,sorc) ; output(sorc,frame[xypos(17,51)],green);
321   end;

323 36:   begin
324   {
325       Get 4th internal cal position.

327   }
328       Sample74 := Sdiga[7] ;
329       sorc := 'Sample 74 ' ; output(sorc,frame[xypos(17,61)],white);
330       str(sample74:5,sorc) ; output(sorc,frame[xypos(17,71)],green);
331       writeln(1st,days:5,hrs:5,mins:3,secs:6:2,spacecount:4,
332       spacelook:10,sample9:10,sample10:10,sample71:10,sample74:10);
333       spacelook := 0 ;
334       spacecount := 0 ;
335   end;
336 else   begin end;
337 end; { of case }
338 end; { of procedure }

344 PROCEDURE DISPLAYACRO ;
345 VAR I: INTEGER;
346 BEGIN
347   txtfile := 'sorc.txt' ;
348   assign(txt,txtfile);
349   reset(txt);
350   att := 15 ;
351   i := 80 ;
352   while not eof(txt) do
353   begin
354     readln(txt,sorc);
355     sorc := sorc + '          ' ;
356     putout(sorc,frame[i],att);

```

Listing of: SCANCHK.PAS

```

357  i := i + 80 ;
358  end;
359  close (txt);

361  END;
362  procedure forwardjump ;
363  var ipos : integer ;
364  begin
365      ipos := filepos(f1) ;
366      ipos := ipos + 100 ;
367      seek(f1,ipos) ;

369  end ;

372  { BEGINNING OF THE MAIN PROGRAM-----}

376  BEGIN
377      ESIPL := 'OFF' ;
378  WRITELN(' ENTER SPACE CRAFT ID (9 OR 10) ');
379  READLN(IDNUM);
380  IF IDNUM = 9 THEN ASSIGN(FL,'NOAA9.DAT')
381  ELSE ASSIGN(FL,'NOAA10.DAT');
382  RESET(FL);
383  STORINT[0] := SCRINT[0] ;           { SAVE PRINT SCREEN VECTOR }
384  STORINT[1] := SCRINT[1] ;
385  SCRINT[0] := DFS(FRSTAT) ;         { PLACE NEW VECTOR TO OUR ROUTINE }
386  SCRINT[1] := CSEG ;
387  NEWSCREEN ;                       { SET UP 80 X 50 DISPLAY SCREEN }
388  DISPLAYACRO ;                     { DISPLAY ACRONYMS ON SCREEN }
389  STATPR := 0;  linenum := 0;  charnum:= 0;
390  spacecount := 0 ;
391  spacelook := 0 ;
392  MINORFRAMENUM := -1 ;
393  WRITELN(LST,' SCAN CHECK FOR NOAA-',IDNUM:2);
394  repeat

396  REEP:

398      key := cntlb ;
399      if (key = 2 ) then goto stop ;
400      if (key = 18 ) then displayacro ;
401      ( if (key = 6 ) then forwardjump ; )
402      if (key = 11 ) then
403          begin
404              repeat
405                  key := cntlb ;
406                  if key = 7 then goto reep ;
407                  until key = 11 ;

```

## Listing of: SCANCHK.PAS

```
408         rewrite(f1) ;
409         goto stop ;
410     end;
411     if (key = 16 ) then
412     repeat
413         key := cnt1b ;
414     until key = 7 ;

418         DIVY ; { EXTRACT USEFUL DATA FROM TIP MINOR FRAME MN }
419         if (minorframenum >319) then goto bottom ;

421 { Process scan position data }
422     Scanpos ;

424 { Output minorframe number }
425     str(minorframenum:5,sorc);
426     putout(sorc,frame[xypos(42,15)],white);
427     bottom:

429     until eof(f1) ;
430 STOP:   SCRINT[0] := STORINT[0] ;
431         SCRINT[1] := STORINT[1] ;
432         CLOSE(FL);
433         OLDSCREEN ;
434     END. { of socc disp program }
```



## APPENDIX C - ESMCAL

ESMCAL (ERBS scanner mam calibration program) is used to determine the stability of the scanner detectors during solar transit. Sensor and temperature data are tabulated by this program and plots of the short wave and total channels are generated for the complete solar calibration time span. From this data, a time history was established to determine the stability of the detectors.

PURPOSE : ESMCAL IS THE SHORTWAVE CHECK ONE  
STABILITY OF THE TOTAL AND SHORTWAVE ERBS  
SCANNER CHANNEL DURING A SOLAR CALIBRATION  
ESMCAL IS ERBS SCANNER MAM CALIBRATION PROGRAM.  
THIS GETS THE SOLAR CAL DATA FOR ALL THE THREE  
RADIOMETRIC CHANNELS DURING A SOLAR TRANSIT.  
THIS GETS A TABULATED FORM OF PRINT OUT OF THE  
SENSORS DATA AND A TABULATED FORM OF TEMPERATURE  
IN ENGINEERING UNITS. ALSO GETS 2 PLOTS OF SSCH,  
STCH FOR THE WHOLE PERIOD DURING A SOLAR CALIBRATION.  
FROM THIS DATA, TIME HISTORY WAS DONE TO SEE  
IF THE RESPONSE OF THE DETECTORS WERE CONSTANT  
OR CHANGING

LANGUAGE : FORTRAN-5

PROCEDURE: TYPE :  
-ERBS, ESMCALJ, DISKF, SMMARY

```
.PROC,ERBS,DISKF,SMMARY,FIL=#DATA.  
REFORM,FIL.  
ROUTE,SCR,DC=IN,ST=RHA.  
REVERT.ESMCAFR  
.DATA.  
/JOB  
COMND,T2000,CM77200.  
/USER  
/CHARGE  
GET,USER1.  
DELIVER.BIN15SU NATARAJAN  
GET,TAPE1=DISKF.  
REWIND,TAPE1.  
ATTACH,LARCGOS/UN=LIBRARY,NA.  
GET,CABLIB5/UN=UTIL.  
GET,ESMCAL.  
FTNS,I=ESMCAL.  
LDSET,LIB=CABLIB5.  
LDSET,LIB=LARCGOS,PRESETA=NGINF.  
LGO.  
REWIND,TAPE11.  
REPLACE,TAPE11=SMMARY.  
ROUTE,TAPE11,DC=LP.  
REPLACE,SAVPLT.  
PLOT.CALPOST,11(PAGE=19,XM=.7,FSH=19)  
CONT.//600A PERFORATED 8.5 X 11 ROLL PAPER  
CONT.BLANK PAPER BLACK INK GREEN INK RED INK  
CONT.PEN 1 - BLACK LEROY 3  
CONT.PEN 2 - GREEN LEROY 3  
CONT.PEN 3 - RED LEROY 3  
CONT.PLEASE FOLD THESE PLOTS//  
DAYFILE,ERBSOK.  
REPLACE,ERBSOK.  
EXIT.  
DAYFILE,ERBSER.  
REPLACE,ERBSER.
```

PROGRAM ESMCAL 74/060 OPT=1,RUIND=A/S/M/-D,-DS 87/05/21.08.21.17  
DO=-LONG/-OT,ARG=COMMON/-FIXED,CS=USER/-FIXED,DB=TR/-SB/-SL/-ER/-ID/-PMD/-ST,-AL,PL=5000  
FTN5,I=ESMCAL,L=LF.

PROGRAM ESMCAL

WRITTEN BY : SUDHA NATARAJAN  
SYSTEM : NOS CDC  
PROJECT : ERBE  
ST SYSTEMS CORPORATION  
HAMPTON VA-23666.

THIS IS ERBS SANNER MAM CALIBRATION PROGRAM.  
THIS PROGRAM IS USED TO OBTAIN SOLAR CALIBRATION DATA FOR ALL C  
THE THREE RADIOMETRIC CHANNEL DURING A SOLAR TRANSIT. C  
THE SHORTWAVE AND TOTAL CHANNELS USE THE MAM (MIRROP C  
ATTENUATOR MOSAIC) AS A PRIMARY CALIBRATION DURING A SOLAR C  
VIEW \*  
\*  
ROUTINES USED : BUFFER IN,SPREAD  
SUBROUTINES CALLED : TIMM,CTEU,CHPLT

CC \*  
DLT60L SLCH SAMPLE 60 OUTPUT \*  
DLT60S SSCH SAMPLE 60 OUTPUT \*  
DLT60T STCH SAMPLE 60 OUTPUT \*  
DLT73L SLCH SAMPLE 73 OUTPUT \*  
DLT73S SSCH SAMPLE 73 OUTPUT \*  
DLT73T STCH SAMPLE 73 OUTPUT \*  
IBUF ERBS RECORD IN LARC FORMAT \*  
IBYTES ERBS RECORD WHEN 'SPREAD' ROUTINE IS USED \*  
IDAY THE DAY NUMBER OF ORBITAL DATA BEING ANALYSED \*  
IECHO COMMAND ECHO \*  
IYEAR YEAR THE DATA IS RECORDED \*  
K INDEX OF THE TEMP. REF VOLTAGE IN EACH MF \*  
L INDEX OF THE MAM TEMP. IN EACH MF \*  
M INDEX IF THE TOTAL BLACK BODY TEMP. IN EACH MF \*  
N INDEX OF THE DETECTOR TEMP. IN EACH MF \*  
SCH3 AVERAGE OF SSCH SAMPLE 3 OF 4 SCANS \*  
SCH60 AVERAGE OF SSCH SAMPLE 60 OF 4 SCANS \*  
SCH73 AVERAGE OF SSCH SAMPLE 73 OF 4 SCANS \*

ORIGINAL PAGE IS  
OF POOR QUALITY

ORIGINAL PAGE IS  
OF POOR QUALITY

```

41 C SLCH3 AVERAGE OF SLCH SAMPLE 3 OF 4 SCANS *
42 C SLCH60 AVERAGE OF SLCH SAMPLE 60 OF 4 SCANS *
43 C SLCH73 AVERAGE OF SLCH SAMPLE 73 OF 4 SCANS *
44 C SLDETT LONGWAVE DETECTOR TEMP IN ENG. UNITS *
45 C SSDETT SHORTWAVE DETECTOR TEMP IN ENG. UNITS *
46 C SSMAMT SHORTWAVE MAM TEMP. IN ENG.UNITS *
47 C SSM8T SHORTWAVE MAM BAFFLE TEMP IN ENG.UNITS *
48 C ST8BT TOTAL BLACK BODY TEMP IN ENG.UNITS *
49 C STDETT TOTAL DETECTOR TEMP IN ENG UNITS *
50 C STMAMT TOTAL MAM TEMP IN ENG. UNITS *
51 C ST8BT TOTLA MAM RAFFLE TEMP IN ENG UNITS *
52 C SLCH3 AVERAGE OF STCH SAMPLE 3 OF 4 SCANS *
53 C SLCH60 AVERAGE OF STCH SAMPLE 60 OF 4 SCANS *
54 C SLCH73 AVERAGE OF STCH SAMPLE 73 OF 4 SCANS *
55 C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC *
56 C *
57 C COMMON/MAM/SSMAMT(130),SSM8T(130),STMAMT(130),ST8BT(130),
58 C *ST8BT(130),SSDETT(130),SLDETT(130),STDETT(130),STRV(130)
59 C COMMON/TIM/JDAY(130),JHR(130),JMIN(130),JSEC(130)
60 C COMMON/FTIM/FYEAR,FDAY,FHR,FMIN,FSEC,FHRE,FMINF,FSECE
61 C INTEGER ACCM3,ACCM60,ACCM73,ACCM3,RCCM60,BCCM73,DCCM3,DCCM60,
62 C *DCCM73
63 C DIMENSION IBUF(402),IBYTES(3015),SCH3(130),SCH60(130),
64 C *SCH73(130),SLCH3(130),SLCH60(130),SLCH73(130),STCH3(130),
65 C *STCH60(130),STCH73(130),X(130),DLT60S(130),DLT73S(130),
66 C *DLT60L(130),DLT73L(130),DLT60T(130),DLT73T(130),IFCHM(130)
67 C IP=0
68 C IPDS3=58
69 C IPDS60=536
70 C IPU573=682
71 C MAMT=700
72 C IT8B=716
73 C IDET=719
74 C ITRV=710
75 C
76 C BEGIN BUFFERING IN THE RECORD UNTIL A VALID RECORD IS
77 C ENCOUNTERED.
78 C USE SPREAD ROUTINE TO STORE 8 BITS IN CDC 60 BIT WORD.
79 C CALL TO SUBROUTINE 'TIMM' TO GET THE TIME
80 C
81 C 1 BUFFER IN(1,1)(IBUF(1),IBUF(402))
82 C 2 IF(UNIT(1)2,500,500

```

```

83 2 BUFFER IN(1,1)(IRUF(1),IRUF(402))
84 IF(UNIT(1))22,500,500
85 K=LENGTH(1)
86 CALL SPREAD(IBUF(1),IBYTES(1),8,60,402,N)
87 CALL TIMM(IBYTES,ISEC,IYEAR,KHR,KMIN,KSEC,IDAY)
88 C
89 C INITIALISE THE ACCUMULATORS TO ZERO
90 C
91 ACCM3=0
92 ACCM60=0
93 ACCM73=0
94
95 BCCM3=0
96 BCCM60=0
97 BCCM73=0
98 C
99 DCCM3=0
100 DCCM60=0
101 DCCM73=0
102 NSMSM=0
103 NSMSB=0
104 NSMTM=0
105 NSMT8=0
106 NSMTBB=0
107 NSMSDT=0
108 NSMLDT=0
109 NSMTDT=0
110 NSMTRV=0
111 IP=IP+1
112 C
113 C PROCESS EACH SCAN
114 C
115 DO 200 I=1,4
116 IPT3= (I-1)*752 + IP053
117 IPT60 = (I-1)*752 + IP0560
118 IPT73 = (I-1)*752 + IP0573
119 C
120 L= (I-1)*752 + MAMT
121 M= (I-1)*752 + ITBB
122 N= (I-1)*752 + IDET
123 K=(I-1)*752 + ITRV
124 C

```

GET THE DATA OF SAMPLE 3,60,73 OF THE 3 DETECTORS

```

126 IA=SHIFT(IBYTES(IPT3+1),-4).AND.15
127 IB=SHIFT(IBYTES(IPT3),+4)
128 ISCH3=IB.OR.IA
129 IA=IBYTES(IPT3+1).AND.15
130 ILCH3=SHIFT((IA),+8).OR.IBYTES(IPT3+2)
131 IA=SHIFT(IBYTES(IPT3+4),-4).AND.15
132 IB=SHIFT(IBYTES(IPT3+3),+4)
133 ITCH3=IB.OR.IA
134
135 IA=SHIFT(IBYTES(IPT60+1),-4).AND.15
136 IB=SHIFT(IBYTES(IPT60),+4)
137 ISCH60=IB.OR.IA
138 IA=IBYTES(IPT60+1).AND.15
139 ILCH60=SHIFT((IA),+8).OR.IBYTES(IPT60+2)
140 IA=SHIFT(IBYTES(IPT60+4),-4).AND.15
141 IB=SHIFT(IBYTES(IPT60+3),+4)
142 ITCH60=IB.OR.IA
143
144 IA=SHIFT(IBYTES(IPT73+1),-4).AND.15
145 IB=SHIFT(IBYTES(IPT73),+4)
146 ISCH73=IB.OR.IA
147 IA=IBYTES(IPT73+1).AND.15
148 ILCH73=SHIFT((IA),+8).OR.IBYTES(IPT73+2)
149 IA=SHIFT(IBYTES(IPT73+4),-4).AND.15
150 IB=SHIFT(IBYTES(IPT73+3),+4)
151 ITCH73=IB.OR.IA
152
153
154
155 GET THE TEMPERATURE DATA
156 GET,SHORTWAVE MAM,SHORTWAVE MAM BAFFLE TEMP.,TOTAL MAM TEMP,
157 TOTAL BAFFLE TEMP.,TOTAL BLACK BODY TEMP.,SHORTWAVE DETECTOR
158 TEMP.,LONGWAVE DETECTOR TEMP.,TOTAL DETECTOR TEMP.
159
160 IA=SHIFT(IBYTES(L),+4)
161 IB=SHIFT(IBYTES(L+1),-4).AND.15
162 ISMAM=IA.OR.IB
163 IA=IBYTES(L+1).AND.15
164 ISMBT=SHIFT((IA),+8).OR.IBYTES(L+2)
165 IA=SHIFT(IBYTES(L+3),+4)
166 IB=SHIFT(IBYTES(L+4),-4).AND.15

```

C

C

C

C

C

C

C

C

C

ORIGINAL PAGE IS  
OF POOR QUALITY

```

167 ITMAM = IA .OR. IB
168 IA = IRYTES(L+4) .AND. 15
169 ITMBT = SHIFT((IA),+8) .OR. IRYTES(L+5)
170 IA = IRYTES(M) .AND. 15
171 ITBBT = SHIFT((IA),+8) .OR. IRYTES(M+1)
172 IA = IRYTES(N) .AND. 15
173 ISDET = SHIFT((IA),+8) .OR. IRYTES(N+1)
174 IA = SHIFT(IRYTES(N+2),+4)
175 IB = SHIFT(IRYTES(N+3),-4) .AND. 15
176 ILDET = IA .OR. IB
177 IA = IRYTES(N+3) .AND. 15
178 ITDET = SHIFT((IA),+8) .OR. IRYTES(N+4)
179 IA=IRYTES(K) .AND. 15
180 ISTRV=SHIFT((IA),+8).OR.IRYTES(K+1)
181
182 ACCM3 = ACCM3 + ISCH3
183 ACCM60 = ACCM60 + ISCH60
184 ACCM73 = ACCM73 + ISCH73
185
186 BCCM3 = BCCM3 + ILCH3
187 BCCM60 = BCCM60 + ILCH60
188 BCCM73 = BCCM73 + ILCH73
189
190 DCCM3 = DCCM3 + ITCH3
191 DCCM60 = DCCM60 + ITCH60
192 DCCM73 = DCCM73 + ITCH73
193 NSMSM = NSMSM + ISMAW
194 NSMSB = NSMSB + ISMBT
195 NSMTM = NSMTM + ITMAM
196 NSMTB = NSMTB + ITMBT
197 NSMTBB = NSMTBB + ITBBT
198 NSMSDT = NSMSDT + ISDET
199 NSMLDT = NSMLDT + ILDET
200 NSMTDT = NSMTDT + ITDET
201 NSMTRV=NSMTRV + ISTRV
202
203 CONTINUE
204 JDAY(IP)=IDAY
205 JHR(IP)=KHR
206 JMIN(IP)=KMIN
207 JSEC(IP)=KSEC
208

```

C

C

C

C

200

C

209 C GET THE AVERAGE SAMPLE 3,60,73 OF THE DETECTORS OF 4 SCANS

210 C  
211 SCH3(IP) = FLOAT(ACCM3) / 4.  
212 SCH60(IP) = FLOAT(ACCM60) / 4.  
213 DLT60S(IP) = SCH60(IP) - SCH3(IP)  
214 SCH73(IP) = FLOAT(ACCM73) / 4.  
215 DLT73S(IP) = SCH73(IP) - SCH3(IP)  
216 C

217 SLCH3(IP) = FLOAT(BCCM3) / 4.  
218 SLCH60(IP) = FLOAT(BCCM60) / 4.  
219 DLT60L(IP) = SLCH60(IP) - SLCH3(IP)  
220 SLCH73(IP) = FLOAT(BCCM73) / 4.  
221 DLT73L(IP) = SLCH73(IP) - SLCH3(IP)  
222 C

223 STCH3(IP) = FLOAT(DCCM3) / 4.  
224 STCH60(IP) = FLOAT(DCCM60) / 4.  
225 DLT60T(IP) = STCH60(IP) - STCH3(IP)  
226 STCH73(IP) = FLOAT(DCCM73) / 4.  
227 DLT73T(IP) = STCH73(IP) - STCH3(IP)  
228 X(IP) = FLOAT(IP)  
229 C

230 C GET THE AVERAGE TEMP. DATA IN COUNTS OF 4 SCANS

231 C  
232 SSMAMT(IP) = FLOAT(NSMSM) / 4.  
233 SSMBT(IP) = FLOAT(NSMSB) / 4.  
234 STAMT(IP) = FLOAT(NSMTM) / 4.  
235 STMBT(IP) = FLOAT(NSMTB) / 4.  
236 SPHAT(IP) = FLOAT(NSMTRR) / 4.  
237 SDETT(IP) = FLOAT(NSM>DT) / 4.  
238 SLDETT(IP) = FLOAT(NSMLDT) / 4.  
239 STDETT(IP) = FLOAT(NSMTDT) / 4.  
240 STRV(IP) = FLOAT(NSMTRV) / 4.  
241 C

242 IA=IBYTES(728).AND.15  
243 IECHO(IP)=SHIFT((IA),+8).OR.IBYTES(729)

244 C READ ANOTHER RECORD

245 C  
246 C GO TO 2

247 C  
248 C  
249 C CALL TO SUBROUTINE CTEU TO CONVERT TO ENGINEERING UNITS  
250 C



```

251 C
252 C    500 CALL CTFU(IP)
253 C
254 C    WRITE HEADR IN THE OUTPUT FILE
255 C
256 C    WRITE(11,501)IYEAR, IDAY
257 C    FORMAT('1',22X,'FRBS SCANNER RADIOMETRIC TABLE',/33X,
258 C    *'TEST DAY ',/2X,I2,'/',I3,'/',I3,'/10',2X,'MAJOR FRAME',/29X,
259 C    *'POSITIONS',/6X,'TIME',10X,'DET',10X,'SPACE',10X,'MAM',
260 C    *9X,'INT/CAL',10X,'DLTMAM',10X,'DLTINT/CAL',/2X,'HRS/MIN/SECS',
261 C    *6X,'TYPE')
262 C    ILINE=7
263 C
264 C    WRITE RADIOMETRIC DATA IN THE OUTPUT FILE
265 C
266 C    DO 510 IPT=1,IP
267 C    WRITE(11,511)JHR(IPT),JMIN(IPT),JSEC(IPT),SCH3(IPT),SCH60(IPT),
268 C    *SCH73(IPT),DLT60S(IPT),DLT73S(IPT),SLCH3(IPT),SLCH60(IPT),
269 C    *SLCH73(IPT),DLT60L(IPT),DLT73L(IPT),STCH3(IPT),
270 C    *STCH60(IPT),STCH73(IPT),DLT60T(IPT),DLT73T(IPT),TECHN(IPT)
271 C    FORMAT(3X,I2,'/',I2,'/',I2,'/',I2,'/',I2,'/',I2,'/',I2,'/',I2,'/',
272 C    *F7.2,7X,F7.2,7X,F7.2/20X,'SLCH',9X,F7.2,6X,F7.2,7X,
273 C    *F7.2,7X,F7.2,7X,F7.2/20X,'STCH',9X,F7.2,6X,
274 C    *F7.2,7X,F7.2,7X,F7.2,7X,F7.2,7X,I6)
275 C    ILINE=ILINE+3
276 C    IF(ILINE.GT.32)THEN
277 C    WRITE(11,501)IYEAR, IDAY
278 C    ILINE=7
279 C    ENDIF
280 C    CONTINUE
281 C
282 C    WRITE(11,650)IYEAR, IDAY
283 C    FORMAT('1',22X,'EPBS SCANNER TEMPERATURE TABLE',/33X,
284 C    *'TEST DAY ',/2X,I2,'/',I3,'/',I3,'/10',2X,'MAJOR FRAME',/6X,'TIME',/
285 C    *2X,'HRS/MIN/SECS',/2X,'SSMAMT',2X,'SSMBT',3X,'STMAMT',2X,'STMBT',
286 C    *3X,'STABT',3X,'SSDETT',2X,'SLDETT',2X,'STDETT')
287 C    ILINE=7
288 C
289 C    WRITE TEMPERATURE DATA TO THE OUTPUT FILE
290 C
291 C    DO 900 K=1,IP
292 C    WRITE(11,800)JHR(K),JMIN(K),JSEC(K),SSMAMT(K),

```

ORIGINAL PAGE IS  
OF POOR QUALITY

```

293 *SSMBT(K),STMAMT(K),STMBT(K),STBRT(K),
294 *SSDETT(K),SLDETT(K),STOETT(K),
295 ILINE=ILINE+1
296 IF(ILINE.GT.32)THEN
297 WRITE(11,650)IYEAR,I0AY
298 ILINE=7
299 ENDIF
300 CONTINUE
301 FORMAT(3X,I2,/,I2,/,I2,/,I2,/,I2,/,I2,/,I2,/,I2,/,I2,/,I2,/,I2,/,I2,/,I2,/,I2,/,I2,/,I2,/)
302 C
303 CALL TO SUBROUTINE CHPLT TO PLOT THE SPACE,MAM AND INT,CAL
304 LOOK
305 C
306 CALL CHPLT(SCH3,DLT60S,DLT74S,SLCH3,DLT60L,DLT73L,
307 *SCH3,DLT60T,DLT73T,IP,IYEAR,X)
308 GO TO 4
309 RUFFER IN(1,1)(IRUF(1),IRUF(402))
310 IF(UNIT(1))2,4,4
311 STOP
312 END

```

1  
1  
1

---VARIABLE MAP---(LD=A)  
--NAME--ADDRESS --BLOCK--PROPERTIES--TYPE-----SIZE

--NAME--	ADDRESS	BLOCK	PROPERTIES	TYPE	SIZE
ACCM60	706B		INTEGER	INTEGER	130
ACCM73	707B		INTEGER	INTEGER	130
BCCM3	710B		INTEGER	INTEGER	130
BCCM60	711B		INTEGER	INTEGER	130
BCCM73	712B		INTEGER	INTEGER	130
DCCM3	713B		INTEGER	INTEGER	130
DCCM60	714B		INTEGER	INTEGER	130
DCCM73	715B		INTEGER	INTEGER	130
DLT60L	716B		REAL	REAL	130
DLT60S	12500B		REAL	REAL	130
DLT60T	12074B		REAL	REAL	130
DLT73L	13104B		REAL	REAL	130
DLT73S	12702B		REAL	REAL	130
DLT73T	12276B		REAL	REAL	130
FDAY	13306B		REAL	REAL	130
FHR	1B		/FTIM/	/FTIM/	402
	2B		/FTIM/	/FTIM/	3015
	5B		/FTIM/	/FTIM/	402
	3B		/FTIM/	/FTIM/	3015
	6B		/FTIM/	/FTIM/	130
	4B		/FTIM/	/FTIM/	130
	7B		/FTIM/	/FTIM/	130
	8B		/FTIM/	/FTIM/	130
	13743B		INTEGER	INTEGER	130
	13751B		INTEGER	INTEGER	130
	13752B		INTEGER	INTEGER	130
	717B		INTEGER	INTEGER	130
	1541B		INTEGER	INTEGER	130
	13731B		INTEGER	INTEGER	130
	13720B		INTEGER	INTEGER	130
	13510B		INTEGER	INTEGER	130
	13754B		INTEGER	INTEGER	130
	13757B		INTEGER	INTEGER	130
	13762B		INTEGER	INTEGER	130

```

1  SUBROUTINE TIMM
2  DO=-LONG/-DT,ARG= COMMON/-FIXED,CS= USEP/-FIXED,DB=-TR/-SB/-SL/-ER/-ID/-PMD/-ST,-AL,OL=5000
3  FTN5,I=ESMCAL,L=LF.
4
5  C
6  C
7  C
8  C
9  C
10 C
11 C
12 C
13 C
14 C
15 C
16 C
17 C
18 C
19 C
20 C
21 C
22 C
23 C
24 C
25 C
26 C
27 C
28 C
29 C
30 C
31 C
32 C
33 C
34 C
35 C
36 C
37 C
38 C
39 C
40 C

```

```

SUBROUTINE TIMM(IBYTES,ISEC,IYEAR,KHR,KMIN,KSEC,IDAY)
THIS SUBROUTINE GETS THE TIME OF THE ORBITAL DATA
CALLING ROUTINE : ESMCAL
DIMENSION IBYTES(3015)
IA=IBYTES(2).AND.1
IB=SHIFT((IA),+8).OR.IBYTES(172)
IC=SHIFT((IBYTES(173)),-3).AND.31
***GET THE JULIAN DAY
JDAY=SHIFT((IB),+5).OR.IC
* GET THE DAY NUMBER
IF(JDAY.GT.5699 .AND. JDAY .LT.6066)THEN
  IDAY=JDAY-5699
  IYEAR=84
ELSEIF(JDAY.GT.6065 .AND. JDAY.LT.6431)THEN
  IDAY=JDAY-6065
  IYEAR=85
ELSEIF(JDAY.GT.6430 .AND. JDAY .LT. 6796)THEN
  IDAY=JDAY-6430
  IYEAR=86
ELSEIF(JDAY.GT.6795 .AND. JDAY.LT.7161)THEN
  IDAY=JDAY-6795
  IYEAR=87
ENDIF
IO=IBYTES(173).AND.7
IE=SHIFT((ID),+8).OR.IBYTES(174)
IF=SHIFT((IBYTES(175)),-2).AND.63
ISEC=SHIFT((IE),+6).OR.IF
IMIN=ISEC/60
KSEC=ISEC-(IMIN*60)
KHR=IMIN/60
KMIN=IMIN-(KHR*60)
IG=IBYTES(175).AND.3
IMILI=SHIFT((IG),+8).OR.IBYTES(176)
RETURN
END

```

ORIGINAL PAGE IS OF POOR QUALITY

ORIGINAL PAGE IS  
OF POOR QUALITY

SUBROUTINE CTEU 74/860 OPT=1,KUUND= A/ C/ M/-0,-DS FTN 5.1+642  
DD=-LONG/-OT,ARG= COMMON/-FIXED,CS= USER/-FTXED,DR=-TR/-SB/-SL/-ER/-ID/-PMD/-ST,-AL,PL=5000  
FTN5,I=ESMCAL,L=LF.

```

1 SUBROUTINE CTEU(IP)
2
3 C THIS SUBROUTINE GETS THE TEMPERATURE IN ENG.UNITS
4 C USING COUNT CONVERSION EQUATION.
5 C
6 C CALLING ROUTINE : ESMCAL
7 C
8 C
9 C COMMON/MAM/SSMAMT(130),SSMBT(130),STMAMT(130),STMBT(130),
10 C *STRBT(130),SSDETT(130),SLODETT(130),STDFTT(130),STRV(130)
11 C DATA C1,C2,C3,C4,C5,C6/20.0091,.0129966,.025284F-6,
12 C *2.33846E-8,1.67705E-11,3.99253E-15/
13 C DATA B1,B2,B3,B4,B5,B6/12.1752498,1.4411415F-2,5.04308321E-7,
14 C *2.62463814E-11,1.04124505E-14,.022227099E-19/
15 C DATA D1,D2/36,9.766E-4/
16 C DATA S1/2.1106F-3/
17 C DD 100 I=1,IP
18 C
19 C GET THE SHORTWAVE MAM TEMP
20 C
21 C SSMAMT(I) = -C1 +(C2 * SSMAMT(I))-(C3*SSMAMT(I)**2)+
22 C *(C4 * SSMAMT(I)**3) - (C5 * SSMAMT(I)**4)
23 C **+(C6 * SSMAMT(I)**5)
24 C
25 C GET SHORTWAVE MAM BAFFLE TEMP
26 C
27 C SSMBT(I) = -C1 +(C2 * SSMBT(I))-(C3*SSMBT(I)**2)+
28 C *(C4 * SSMBT(I)**3) - (C5 * SSMBT(I)**4)
29 C **+(C6 * SSMBT(I)**5)
30 C
31 C GET TOTAL MAM TEMP
32 C
33 C STMAMT(I) = -C1 +(C2 * STMAMT(I))-(C3*STMAMT(I)**2)+
34 C *(C4 * STMAMT(I)**3) - (C5 * STMAMT(I)**4)
35 C **+(C6 * STMAMT(I)**5)
36 C
37 C GET TOTAL MAM BAFFLE TEMP
38 C
39 C STMBT(I) = -C1 +(C2 * STMBT(I))-(C3*STMBT(I)**2)+
40 C *(C4 * STMBT(I)**3) - (C5 * STMBT(I)**4)

```

ORIGINAL PAGE IS  
OF POOR QUALITY

```

41      ***(C6 * STMBT(I)**5)
42      C
43      GET SHORTWAVE, LONGWAVE, TOTAL DETECTOR TEMP.
44      C
45      SDETT(I) = U1 + D2 * SDETT(I)
46      SLOETT(I) = D1 + D2 * SLOETT(I)
47      STDETT(I) = D1 + D2 * STDETT(I)
48      STRV(I) = -S1 * STRV(I)
49      P = -6.40 / STRV(I)
50      C
51      GET TOTAL BLACK BODY TEMP
52      C
53      STBRT(I) = STBRT(I) * R
54      STBTT(I) = -R1 + (B2 * STBRT(I)) + (B3 * STBTT(I)**2) -
55      *(B4 * STBRT(I)**3) + (B5 * STBTT(I)**4) -
56      *(B6 * STBRT(I)**5)
57      CONTINUE
58      RETURN
59      END

```

--VARIABLE MAP--(LD=A)		--BLOCK--		--PROPERTIES--		--SIZE		--TYPE--		--SIZE		
-NAME-	-ADDRESS	-BLOCK-	-ADDRESS	-BLOCK-	-PROPERTIES-	-SIZE	-TYPE-	-NAME-	-ADDRESS	-BLOCK-	-PROPERTIES-	-SIZE
B1	106R						REAL	I	117B			INTEGER
B2	107R						REAL	IP	1			REAL
B3	110B						REAL	R	120B	DUMMY-ARG		REAL
B4	111R						REAL	SLOETT	1414B	/MAM/		REAL
B5	112B						REAL	SDETT	1212B	/MAM/		REAL
B6	113B						REAL	SSMAMT	OB	/MAM/		REAL
C1	100R						REAL	SSMRT	202B	/MAM/		REAL
C2	101B						REAL	STBRT	1010B	/MAM/		REAL
C3	102R						REAL	STDETT	1616B	/MAM/		REAL
C4	103R						REAL	STMAMT	404B	/MAM/		REAL
C5	104R						REAL	STMRT	606B	/MAM/		REAL
C6	105B						REAL	STRV	2020B	/MAM/		REAL
D1	114B						REAL	S1	116B			REAL
D2	115B						REAL					REAL

SUBROUTINE CHPLT 74/860 OPT=1,ROUND= A/ 9/ M/-D,-DS FTM 5.1+642 #7/05/21. 08.21.17  
 DD=-LONG/-OT,ARG= COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/-ER/-ID/-PMD/-ST,-AL,PL=5000  
 FTN5,I=ESMCAL,L=LF.

```

1 SUBROUTINE CHPLT(SCH3,DLT60S,DLT73S,SLCH3,DLT60L,DLT73L,
2 *STCH3,DLT60T,DLT73T,IP,IYEAR,X)
3
4 C THIS SUBROUTINE PLOTS THE RADIO-METRIC SAMPLES OF SPACE
5 C MAM AND INTERNAL CAL LOCK
6 C
7 C CALLING ROUTINE : ESMCAL
8 C
9 C
10 CHARACTER DETLABL*4
11 COMMON/TIM/JDAY(130),JHR(130),JMIN(130),JSEC(130)
12 COMMON/FTIM/FYEAR,FDAY,FHR,FMIN,FSEC,FHRE,FMINE,FSECF
13 DIMENSION SCH3(IP+2),DLT60S(IP+2),DLT73S(IP+2),X(IP+2)
14 DIMENSION SLCH3(IP+2),DLT60L(IP+2),DLT73L(IP+2),STCH3(IP+2),
15 *DLT60T(IP+2),DLT73T(IP+2)
16 C
17 C SET THE ORIGIN AND SCALE FACTOR FOR THE X AXIS
18 C
19 X(IP+1)=0.
20 X(IP+2)=15.
21 WRITE(11,150)X(IP+1),X(IP+2)
22 FORMAT(2X,2(F7.2))
23 FYEAR=FLOAT(IYEAR)
24 FDAY=JDAY(1)
25 FHR=JHR(1)
26 FMTN=JMIN(1)
27 FSEC=JSEC(1)
28 FHRE=JHR(IP)
29 FMINE=JMIN(IP)
30 FSECF=JSEC(IP)
31 C INITIALISE THE PLOTTING ROUTINE
32 C
33 C
34 CALL PSEUDD
35 CALL LERRY
36 CALL ASCALE(SCH3,8.5,IP,1,10.)
37 CALL ASCALE(DLT60S,8.5,IP,1,10.)
38 CALL ASCALE(DLT73S,8.5,IP,1,10.)
39 CALL CALPLT(2.,1.,-3)
40 C
  
```

ORIGINAL PAGE IS  
OF POOR QUALITY

```

41 C      DRAW THE AXES
42 C
43 CALL AXES(0,0,0,0,15,X(IP+1),X(IP+2),1,4,
44 *RECORDS,14,-7)
45 CALL AXES(0,0,0,90,8,5,SCH3(IP+1),SCH3(IP+2),1,10,
46 *SCPOS-3,14,7)
47 CALL AXES(0,8,5,0,15,X(IP+1),X(IP+2),1,4,1,0,0,1)
48 CALL AXES(15,0,90,8,5,SCH3(IP+1),SCH3(IP+2),1,
49 *10,0,0,-1)
50 CALL NEWPEN (1)
51 C
52 C      PLOT THE SPACE DATA
53 C
54 CALL LINPLT(X,SCH3,IP,1,0,0,1,1)
55 C
56 C      OFFSET THE PLOT WITHIN THE FRAME AND DRAW ANOTHER Y AXES
57 C
58 CALL NEWPEN (2)
59 CALL CALPLT(-1,0,0,-3)
60 CALL AXES(0,0,0,90,8,5,DLT60S(IP+1),DLT60S(IP+2),1,10,
61 *SCPOS-60,14,8)
62 CALL CALPLT(1,0,0,-3)
63 C
64 CALL NEWPEN(2)
65 C
66 C      PLOT SSCH MAM DATA
67 C
68 CALL LINPLT(X,DLT60S,IP,1,0,0,1,2)
69 C
70 C      OFFSET THE PLOT WITHIN THE FRAME AND DRAW 3RD Y AXES
71 C
72 CALL NEWPEN (3)
73 CALL CALPLT(16,0,0,-3)
74 CALL AXES(0,0,0,90,8,5,DLT73S(IP+1),DLT73S(IP+2),1,10,
75 *SCPOS-73,14,8)
76 CALL CALPLT(-16,0,0,-3)
77 C
78 CALL NEWPEN(3)
79 C
80 C      PLOT SSCH INT-CAL DATA
81 C
82 CALL LINPLT(X,DLT73S,IP,1,0,0,1,3)

```

```

83 CALL NEWPEN(1)
84 DETABL='SSCH'
85
86 CALL TO SUBROUTINE ELABL TO LABEL THE PLOT
87
88 CALL ELABL(DETLABL)
89
90 ESTABLISH A NEW REFERENCE POINT FOR THE NEXT FRAME
91
92 CALL NFRAME
93 CALL CALPLT(2,,1,-3)
94 CALL ASCALE(SLCH3,8.5,IP,1,10.)
95 CALL ASCALE(SLCH60,8.5,IP,1,10.)
96 CALL ASCALE(SLCH73,8.5,IP,1,10.)
97 CALL AXES(0,,0,,0,,15,,X(IP+1),X(IP+2),1,,4,,)
98 *RECORDS,,14,-7)
99 CALL AXES(0,,0,,90,,8.5,SLCH3(IP+1),SLCH3(IP+2),1,,10,,)
100 *SCPOS-3,,14,7)
101 CALL AXES(0,,8.5,0,,15,,X(IP+1),X(IP+2),1,,4,,)
102 CALL AXES(15,,0,,90,,8.5,SLCH3(IP+1),SLCH3(IP+2),1,,)
103 *10,,)
104 CALL NEWPEN(1)
105 CALL LINPLT(X,SLCH3,IP,1,0,0,1,1)
106 CALL NEWPEN(2)
107 CALL CALPLT(-1,,0,-3)
108 CALL AXES(0,,0,,90,,8.5,SLCH60(IP+1),SLCH60(IP+2),1,,10,,)
109 *SCPOS-60,,14,8)
110 CALL CALPLT(1,,0,-3)
111 CALL NEWPEN(2)
112 CALL LINPLT(X,SLCH60,IP,1,0,0,1,2)
113 CALL NEWPEN(3)
114 CALL CALPLT(16,,0,-3)
115 CALL AXES(0,,0,,90,,8.5,SLCH73(IP+1),SLCH73(IP+2),1,,10,,)
116 *SCPOS-73,,14,8)
117 CALL CALPLT(-16,,0,-3)
118 CALL NEWPEN(3)
119 CALL LINPLT(X,SLCH73,IP,1,0,0,1,3)
120 CALL NEWPEN(1)
121 DETABL='SLCH'
122 CALL ELABL(DETLABL)
123 CALL NFRAME
124

```



ORIGINAL PAGE IS  
OF POOR QUALITY

```

125 C ESTABLISH A NEW REFERENCE POINT TO PLOT THE TOTAL CHANNEL
126 C
127 CALL CALPLT(2,,1,,-3)
128 C
129 C SCALE THE Y AXES AND STORE THE ORIGIN AND SCALE FACTOR
130 C
131 CALL ASCALE(STCH3,8.5,IP,1,10.)
132 CALL ASCALE(DLT60T,8.5,IP,1,10.)
133 CALL ASCALE(DLT73T,8.5,IP,1,10.)
134 C
135 C DRAW THE AXES
136 C
137 CALL AXES(0.,0.,0.,15.,X(IP+1),X(IP+2),1.,4.,
138 *RECORDS,,14,-7)
139 CALL AXES(0.,0.,90.,8.5,STCH3(IP+1),STCH3(IP+2),1.,10.,
140 *SCPOS-3,,14,7)
141 CALL AXES(0.,8.5,0.,15.,X(IP+1),X(IP+2),1.,4.,,0.0,1)
142 CALL AXES(15.,0.,90.,8.5,STCH3(IP+1),STCH3(IP+2),1.,
143 *10.,,0.0,-1)
144 CALL NEWPEN (1)
145 C
146 C PLOT THE SPACE TOTAL CHANNEL
147 C
148 CALL LINPLT(X,STCH3,IP,1,0,0,1,1)
149 CALL NEWPEN (2)
150 C
151 C OFFSET THE PLOT WITHIN THE FRAME
152 C
153 CALL CALPLT(-1.,0.,-3)
154 CALL AXES(0.,0.,90.,8.5,DLT60T(IP+1),DLT60T(IP+2),1.,10.,
155 *SCPOS-60,,14,6)
156 CALL CALPLT(1.,0.,-3)
157 CALL NEWPEN(2)
158 C
159 C PLOT THE TOTAL MAM DATA
160 C
161 CALL LINPLT(X,DLT60T,IP,1,0,0,1,2)
162 CALL NEWPEN (3)
163 C
164 C OFFSET THE PLOT WITHIN THE FRAME
165 C
166 CALL CALPLT(16.,0.,-3)

```

```

167 CALL AXES(0.,0.,90.,5.,DLT73T(IP+1),DLT73T(IP+2),1.,10.,
168 *SCPOS-73.,14.,8)
169 CALL CALPLT(-16.,0.,-3)
170 CALL NEWPEN(3)
171
172 C DRAW THE TOTAL INT-CAL DATA
173 C
174 CALL LINPLT(X,DLT73T,IP,1,0,0,1,3)
175 CALL NEWPEN(1)
176 DETLABL=STCH1
177
178 C CALL TO SUBROUTINE ELABL TO LABEL THE PLT
179 C
180 CALL ELABL(DFTLABL)
181 CALL NFRAME
182
183 C TERMINATE THE GRAPH
184 C
185 C CALL CALPLT(0.0,0.0,0.0,999)
186 RETURN
187 END
188

```

ORIGINAL PAGE IS  
OF POOR QUALITY

--VARIABLE MAP--(LO=A)		--BLOCK--		--PROPERTIES--		--TYPE--		--SIZE	
--NAME--	--ADDRESS	--BLOCK--	--ADDRESS	--NAME--	--BLOCK--	--PROPERTIES--	--TYPE--	--SIZE	
DETLABL	1026B			FSECE	7B	/FTIM/	REAL		
DLT60L	5	DUMMY-ARG		FYEAR	0B	/FTIM/	REAL		
DLT60S	2	DUMMY-ARG		IP	10	DUMMY-ARG	INTEGER		
DLT60T	8	DUMMY-ARG		IYEAR	11	DUMMY-ARG	INTEGER		130
DLT73L	6	DUMMY-ARG		JDAY	0B	/TIM/	INTEGER		130
DLT73S	3	DUMMY-ARG		JHR	202B	/TIM/	INTEGER		130
DLT73T	9	DUMMY-ARG		JMIN	404B	/TIM/	INTEGER		130
FDAY	1B	/FTIM/		JSEC	606B	/TIM/	INTEGER		130
FHR	2B	/FTIM/		SCH3	1	DUMMY-ARG	REAL		ADJ-ARY
FHRE	5B	/FTIM/		SLCH3	4	DUMMY-ARG	REAL		ADJ-ARY
FMIN	3B	/FTIM/		STCH3	7	DUMMY-ARG	REAL		ADJ-ARY
FMINE	6B	/FTIM/		X	12	DUMMY-ARG	REAL		ADJ-ARY
FSEC	4B	/FTIM/							

SUBROUTINE ELABL(DETLABL)

C  
C  
C  
C  
C  
C

THIS SUBROUTINE LABELS THE PLOTS

CHARACTER DETLABL\*(\*)  
COMMON/FTIM/FYEAR,FDAY,FHP,FMIN,FSEC,FHRE,FMINE,FSECE  
CALL CHARACT(1.0,8.75,10,1ERBS,0.,4.,.2)  
CALL NUMBER(2.0,8.75,10,FYEAR,0.,-1)  
CALL NUMBER(2.5,8.75,10,FDAY,0.,-1)  
CALL NUMBER(3.0,8.75,10,FHR,0.,-1)  
CALL NUMBER(3.5,8.75,10,FMIN,0.,-1)  
CALL NUMBER(4.0,8.75,10,FSEC,0.,-1)  
CALL NUMBER(4.5,8.75,10,FHRE,0.,-1)  
CALL NUMBER(5.0,8.75,10,FMINE,0.,-1)  
CALL NUMBER(5.5,8.75,10,FSECE,0.,-1)  
CALL CHARACT(6.5,8.75,10,DETLABL,0.,4.,.2)  
RETURN  
END

--VARIABLE MAP--(LO=A)

NAME	ADDRESS	BLOCK	PROPERTIES	TYPE	SIZE
DETLABL	1	DUMMY-ARG	CHAR*(*)	REAL	
FDAY	18	/FTIM/	REAL	REAL	
FHR	28	/FTIM/	REAL	REAL	
FHRE	58	/FTIM/	REAL	REAL	
FMIN	38	/FTIM/	REAL	REAL	
FMYNE	68	/FTIM/		REAL	
FSEC	48	/FTIM/		REAL	
FSECE	78	/FTIM/		REAL	
FYEAR	08	/FTIM/		REAL	

--PROCEDURES--(LO=A)

NAME	TYPE	ARGS	CLASS
CHARACT		7	SUBROUTINE
NUMBER		6	SUBROUTINE

ORIGINAL PAGE IS  
OF POOR QUALITY

DO=-LJNG/-OT,ARG= COMMON/-FIXED,CS= USER/-FIXED,OP=-TB/-SB/-SL/-ER/-IN/-PHD/-ST,-AL,PL=5000  
 FTN5,I=ESMCAL,L=LF.

BLOCK DATA COM

1 COMMON/MAM/SSMAMT(130),SSMBT(130),STMAMT(130),STMBT(130),  
 2 \*STBBT(130),SSDETT(130),SLDETT(130),STDETT(130),STRV(130)  
 3 COMMON/TIM/JDAY(130),JHR(130),JMIN(130),JSEC(130)  
 4 COMMON/FTIM/FYEAR,FDAY,FHR,FMIN,FSEC,FHRE,FHRE,FSECE  
 5 FND  
 6

--VARIABLE MAP--(LO=A)		--BLOCK--		--PROPERTIES--		--TYPE--		--SIZE	
-NAME-	-ADDRESS	-BLOCK-	-NAME-	-ADDRESS	-BLOCK-	-PROPERTIES-	-TYPE-	-SIZE-	-SIZE-
FDAY	1B	/FTIM/	JSEC	606B	/TIM/	REAL	INTEGER	130	130
FHR	2B	/FTIM/	SLDETT	1414B	/MAM/	REAL	REAL	130	130
FHRE	5B	/FTIM/	SSDETT	1212B	/MAM/	REAL	REAL	130	130
FMIN	3B	/FTIM/	SSMAMT	OB	/MAM/	REAL	REAL	130	130
FHRE	6B	/FTIM/	SSMRT	202B	/MAM/	REAL	REAL	130	130
FSEC	4B	/FTIM/	STBRT	1010B	/MAM/	REAL	REAL	130	130
FSECE	7B	/FTIM/	STDETT	1616B	/MAM/	REAL	REAL	130	130
FYEAR	OB	/FTIM/	STMAMT	404B	/MAM/	REAL	REAL	130	130
JDAY	OR	/TIM/	STMBT	606B	/MAM/	REAL	REAL	130	130
JHR	202B	/TIM/	STRV	2020B	/MAM/	REAL	REAL	130	130
JMIN	404B	/TIM/				REAL	REAL	130	130

--STATISTICS--

PROGRAM-UNIT LENGTH OB = 0  
 CM LABELLED COMMON LENGTH 32428 = 1698  
 CM STORAGE USED 613008 = 25280  
 COMPILE TIME 0.182 SECONDS

## APPENDIX D - SOLCA

SOLCA determines the solar constant values for the EV channels and SMA channel. The solar constant values from EV channels are compared to SMA once every 2 weeks in order to calibrate/validate earth viewing channels.

FUNCTION : SOLCA GETS THE SOLAR CONSTANT VALUES BY THE EV CHANNELS AND SMA CHANNEL. SMA ARE USED AS STANDARDS. THE SOLAR CONSTANT VALUES FROM EV CHANNELS ARE COMPARED TO SMA ONCE EVERY 2 WEEKS IN ORDER TO CALIBRATE/VALIDATE EARTH VIEWING CHANNELS.

LANGUAGE : FORTRAN-5

PROCEDURE: TYPE:  
-NSCAN, NSCPROC

.PROC, NSCAN.

NOTE./THIS IS NONSCANNER PROCEDURE TO GET SOLAR SUMMARY/

GET, SOLCA.

FTNS, I=SOLCA, L=LF.

GET, TAPE1=NDATA.

GET, TAPE2=TERBS.

GET, TAPE3=N10DATA.

LGO.

NOTE./TO GET A PRINTED OUTPUT OF SOLAR-CAL SUMMARY/

NOTE./ROUTE, TAPE4, DC=LP (FOR NOAA-9 SUMMARY )/

NOTE./ROUTE, TAPE5, DC=LP. (FOR ERBS SUMMARY)/

NOTE./ROUTE, TAPE7, DC=LP. (FOR NOAA-10 SUMMARY)/

REVERT. NSCAN.

PROGRAM SOLCA

AUTHOR : SUDHA NATARAJAN  
SYSTEM : NOS CDC  
PROJECT : ERBE

THIS SOFTWARE IS USED TO GET THE 'SOLAR CONSTANT' VALUES  
DETERMINED SIMULTANEOUSLY BY THE EARTH VIEWING AND  
SOLAR MONITOR ASSEMBLY(SMA) CHANNELS. SMA VALUES ARE USED  
AS STANDARDS. THE 'SOLAR CONSTANT' VALUES FROM EV CHANNELS  
ARE COMPARED TO SMA ONCE EVERY 2 WEEKS IN ORDER TO  
CALIBRATE/VALIDATE EARTH VIEWING CHANNELS.

CC  
A ARRAY OF THE RADIO-METRIC CHANNEL DATA C  
CALIBRATION DATE C  
TIME OF SPACE LOOK OF THE EARTH CHANNELS C  
MED-TOT AVE LIMIT TEMP(SPACE LOOK) C  
MED-SW AVE. LIMIT TEMP(SPACE LOOK) C  
WIDE-TOT AVE. LIMIT TEMP(SPACE LOOK) C  
WIDE-SW AVE. LIMIT TEMP(SPACE LOOK) C  
ARRAY OF RADIO-METRIC CHANNEL DATA OF SOLAR VIEW C  
STAN. DEV OF THE RAD.CHANNELS DURING SOLAR VIEW C  
SOLAR VIEW TIME OF THE EARTH VIEWING CHANNELS C  
THE DAY NUMBER C  
SUN LOOK OF THE SOL-MON DURING SHUTTER OPEN C  
AVE LIMIT TEMP OF MED-TOT DURING SUN LOOK C  
AVE LIMIT TEMP OF MED-SW DURING SUN LOOK C  
AVE LIMIT TEMP OF WID-TOT DURING SUN LOOK C  
AVE LIMIT TEMP OF WID-SW DURING SUN LOOK C  
SLRPT TEMP OF MED-SW DURING SUN LOOK C  
SLRPT TEMP OF WID-SW DURING SUN LOOK C

ORIGINAL PAGE IS  
OF POOR QUALITY

ORIGINAL PAGE IS  
OF POOR QUALITY

```

41 C BNMHT HEAT SINK TEMP OF SOLAR MON DURING SUN LOOK C
42 C BNMAT APER-TEMP OF SOLAR MON DURING SUN LOOK C
43 C RNMBT BAFFLE TEMP OF SOLAR MON DURING SUN LOOK C
44 C SNANG(1) SUN ANGLE OF THE EARTH CHANNELS DURING SUN LOOK C
45 C SNANG(2) SUN ANGLE OF SOLAR MON DURING SUN LOOK C
46 C
47 C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
48 C
49 C
50 C
51 C COMMON/COEFF/RESTN(6),AREA(5),SWTRN(2)
52 C ENTER THE FLIGHT MODEL
53 C
54 C
55 C
56 C PRINT 100
57 C FORMAT(10X,'1-NOAA-9'/10X,'2-ERBS'/10X,'3-NOAA-10'/10X,
58 C *'ENTER THE FLIGHT MODEL')
59 C READ *,IDF
60 C FORMAT(A8,1X,I4,1X,A8,1X,6(F6.1,1X)/6(F4.1,1X)/9(F4.1,1X)/
61 C *I3,1X,A8,1X,6(F6.1,1X)/6(F4.1,1X),A8,1X/9(F4.1,1X),F4.2,1X,F4.2)
62 C
63 C ENTER THE ACTIVE HEATER RESISTANCE,AREA OF THE PRIMARY
64 C APERTURE,SHORTWAVE CHANNEL DOME FILTER TRANSMISSION.
65 C
66 C
67 C ENTER THE RESISTANCE,AREA AND TRANSMISSION OF MED-TOT,
68 C MED-SW,MID-TOT,WID-SW,SOLAR CHANNELS
69 C
70 C
71 C PRINT *,' ENTER THE MEDIUM TOTAL CHANNEL RESISTANCE '
72 C READ *,RESTN(1)
73 C PRINT *,' ENTER THE AREA FOR THE MEDIUM TOTAL CHANNEL '
74 C READ *, AREA(1)
75 C PRINT *,' ENTER THE MEDIUM SHORTWAVE CHANNEL RESISTANCE '
76 C READ *,RESTN(2)
77 C PRINT *,' ENTER THE AREA FOR THE MEDIUM SHIRTWAVE CHANNEL '
78 C READ *,AREA(2)
79 C PRINT *,' ENTER THE WIDE TOTAL CHANNEL RESISTANCE '
80 C READ *,RESTN(3)
81 C PRINT *,' ENTER THE AREA FOR THE WIDE TOTAL CHANNEL '
P2 C READ *,AREA(3)

```



```

83 PRINT *, ' ENTER THE WIDE SHORTWAVE CHANNEL RESISTANCE '
84 READ *, RESTN(4)
85 PRINT *, ' ENTER THE AREA FOR THE WIDE SHORTWAVE CHANNEL '
86 READ *, AREA(4)
87 PRINT *, ' ENTER THE SOLAR MONITOR CHANNEL RESISTANCE '
88 READ *, RESTN(5)
89 RESTN(6) = RESTN(5)
90 PRINT *, ' ENTER THE AREA FOR THE SOLAR MONITOR CHANNEL '
91 READ *, AREA(5)
92 PRINT *, ' ENTER THE TRANSMISSION FOR THE MEDIUM SHORTWAVE CH '
93 READ *, SWTPN(1)
94 PRINT *, ' ENTER THE TRANSMISSION FOR THE WIDE SHORTWAVE CH '
95 READ *, SWTPN(2)
96 C
97 C
98 C
99 C
100
101
102
103

```

CALL TO SUBROUTINE PROC TO PROCESS THE RAW DATA TO IRRADIANCE  
CALL PROC(IDF)

STOP  
END

--VARIABLE MAP--(LO=A)

--NAME--	ADDRESS	BLOCK	PROPERTIES	TYPE	SIZE
AREA	6B		/COEFF/	REAL	5
IDF	361B			INTEGER	
PESTN	0B		/COEFF/	REAL	6
SWTRN	13B		/COEFF/	REAL	2

--PROCEDURES--(LO=A)

--NAME--	TYPE	ARGS	CLASS
PROC		1	SUBROUTINE

DD=-LONG/-DT,ARG= COMMON/-FIXED,CS= USER/-FIXED,DB=-T8/-S8/-SL/-ER/-ID/-PHD/-ST,-AL,PL=5000

ORIGINAL PAGE IS  
OF POOR QUALITY

```

SUBROUTINE PROC(IDF)
C
C
C
C
C
C
C
COMMON/COEFF/RESTN(6),AREA(5),SWTRN(2)
COMMON/AVAR/AMTSP,BMTSP,AMSPT,BMSPT,AWTSP,BWTSP,AWSP,BWSP
DIMENSION A(6),B(6),C(6),D(6),E(6),F(5),ASD(6),
* RSD(6),SNANG(2),PRDCT(5)
CHARACTER SPMND*7,ADATE*8,ATIME*8,BTIME*8,BSTME*8,ANS*1
IREC=0
1 READ(IDF,110,END=4010,ERR=99,IOSTAT=IOS)
*ADATE,INO,ATIME,(A(I),I=1,6),
* (ASD(I),I=1,6),AMTSP,AMSSP,AWTSP,AWSSP,AMSPT,AWSP,ANMHT,
* ANMAT,ANMBT,IDAY,BTIME,(B(I),I=1,6),(BSD(I),I=1,6),
* BSTME,BMTSP,BMSSP,BWTSP,BWSSP,BMSPT,BWSP,BNMHT,
* BNMAT,ANMAT,(SNANG(I),I=1,2)
110 FORMAT(A8,1X,I4,1X,A8,1X,6(F6.1,1X)/6(F4.1,1X)/9(F4.1,1X)/
* I3,1X,A8,1X,6(F6.1,1X)/6(F4.1,1X),A8,1X/9(F4.1,1X),F4.2,1X,F4.2)
PRINT 111,ADATE,INO,ATIME,(A(I),I=1,6),(ASD(I),I=1,6),
*AMTSP,AMSSP,AWTSP,AWSSP,AMSPT,AWSP,ANMHT,ANMAT,ANMBT,
*IDAY,BTIME,(B(I),I=1,6),(BSD(I),I=1,6),BSTME,BMTSP,BMSSP,
* BWTSP,BWSSP,BMSPT,BWSP,ANMAT,ANMBT,
111 FORMAT(A8,I4,A8,6(F6.1),6(F4.1)/9(F4.1)/I3,A8,6(F6.1)/
* 6(F4.1),A8/9(F4.1))
C
C
C
CONVERT THE COUNTS TO POWER DURING SPACE AND SUN LOOK
APPLY THE RESISTANCE,AREA,TRANSMISSION TO
C FIND THE MEASURED SOLAR IRRADIANCE FROM THE EARTH
VIEWING CHANNEL
C
C
C
DO 30 I=1,6
C(I) = (A(I)/819.1)**2/RESTN(I)
D(I) = (B(I)/819.1)**2/RESTN(I)
E(I) =ABS(C(I)-D(I))
PRINT 115,C(I),D(I),F(1)

```

```

41 115 FORMAT(F8.7,2X,F8.7,2X,F8.7)
42 30 CONTINUE
43 C
44 C
45 C
46 C
47 DO 40 I=1,4
48 F(I) = E(I)/AREA(I)
49 PRINT 116,F(1)
50 FORMAT (F7.2)
51 CONTINUE
52 40
53 C
54 C
55 CALCULATE THE ACTUAL IRRADIANCE FROM THE SOLAR
56 MONITOR CHANNEL
57 C
58 F(5) = E(5) + E(6)
59 F(5) = F(5)/AREA(5)
60 PRINT 117,F(5)
61 FORMAT (F7.2)
62 117
63 C
64 C
65 C
66 C
67 C
68 C
69 C
70 118 APPLY TRANSMISSION CORRECTION ON THE SW CHANNELS
71 C
72 C
73 F(2) = F(2)/SWTRN(1)
74 F(4) = F(4)/SWTRN(2)
75 PRINT 118,F(2),F(4)
76 FORMAT (F7.2,2X,F7.2)
77 GO TO (120,130,120)IDF
78 C
79 CALL TO SUBROUTINE NOA9 TO DO SOME SPECIFIC PROCESSING
80 FOR TIROS
81 C
82 CALL NOA9(F,SPMDD,IDF,ITAPE)
83 GO TO 150
84 C
85 CALL TO ROUTINE ERBS TO DO SOME SPECIFIC PROCESSING
86 FOR ERBS
87 C
88 CALL ERBS(F,SPMDD,ITAPE)
89 130

```

ORIGINAL PAGE IS  
OF POOR QUALITY

```

83 C
84 C
85 C
86 C
87 C
88 C
89 C
90 C
91 C
92 C
93 C
94 C
95 C
96 C
97 C
98 C
99 C
100 C
101 C
102 C
103 C
104 C
105 C
106 C
107 C
108 C
109 C
110 C
111 C
112 C
113 C
114 C
115 C
116 C
117 C
118 C
119 C
120 C
121 C
122 C
123 C
124 C

CALCULATE THE JULIAN DAY
GET THE SUN DISTANCE USING AN ALGORITHM

IF(ADATE(7:8).EQ.'84')JDAY=IDAY+2445700
IF(ADATE(7:8).EQ.'85')JDAY=IDAY+2446066
IF(ADATE(7:8).EQ.'86')JDAY=IDAY+2446431
IF(ADATE(7:8).EQ.'87')JDAY=IDAY+2446796

JULIAN DAY REFERRED TO 1/1/2000 AS 2451545

N=JDAY-2451545
G=357.528+0.9856003*FLOAT(N)
SNDIST=1.00014-0.01671*CDSD(G)-0.00014*CDSD(2*G)
AUCOR=1/(SNDIST)**2

CORRECTION FOR ONE ASTRONOMICAL UNIT

DO 60 I=1,5
F(I) =F(I)/AUCOR
PRINT 118,F(1),F(5)
CONTINUE

APPLY SUN ANGLE CORRECTION ON EV CHANNELS

DO 65 I=1,4
F(I) = F(I)/COSD(SNANG(1))
PRINT 118,F(1),F(4)
CONTINUE

APPLY SUN ANGLE CORRECTION ON SMA CHANNEL

F(5) = F(5)/COSD(SNANG(2))
PRINT 119,F(5)
FORMAT (F7.2)
DO 70 I=1,4
PRODUCT(I) = AREA(I) * RESTN(I)
CONTINUE
PRODUCT(5) = AREA(5) * RESTN(6)
PRODUCT(2) = PRODUCT(2) * SWTRN(1)

```

ORIGINAL PAGE IS  
OF POOR QUALITY

```

125      PRODUCT(4) = PRODUCT(4) * SWTRN(2)
126
127      CALL THE HEADR ROUTINE TO WRITE THE MAIN HEADR
128
129      CALL HEADR(ITAPE,SPMOD,PRODDT,IREC)
130
131      WRITE THE SOLAR FLUX VALUES OF THE EARTH CH TO
132      THE OUTPUT FILE
133
134      WRITE (ITAPE,2800)ADATE,ATIME,A(1),ASD(1),AMTSP,
135      *A(2),ASD(2),AMSSP,A(3),ASD(3),AMTSP,A(4),ASD(4),AMSSP
136      FORMAT(1X,A8,2X,A8,5X,F6.1,2X,F4.1,4X,F4.1,5X,F6.1,2X,F4.1,4X,
137      *F4.1,3X,F6.1,2X,F4.1,4X,F4.1,6X,F6.1,3X,F4.1,5X,F4.1)
138      IF(R(1).LT.25.)GO TO 102
139      WRITE (ITAPE,2900)IDAY,BTIME,B(1),BSD(1),BMTSP,B(2),BSD(2),
140      *BMSSP,B(3),RSD(3),BWTSP,B(4),BSD(4),BMSSP
141      GO TO 103
142
143      WRITE(ITAPE,2901)TDAY,RTIME,BSD(1),BMTSP,B(2),BSD(2),
144      *RMSSP,R(3),RSD(3),RWTSP,B(4),BSD(4),RMSSP
145      FORMAT (3X,I3,5X,A8,5X,F6.1,2X,F4.1,4X,F4.1,5X,F6.1,2X,
146      *F4.1,4X,F4.1,3X,F6.1,2X,F4.1,4X,F4.1,6X,F6.1,3X,F4.1,5X,F4.1)
147      FORMAT (3X,I3,5X,A8,3X,10FF SCALE,1X,F4.1,4X,F4.1,5X,F6.1,2X,
148      *F4.1,4X,F4.1,3X,F6.1,2X,F4.1,4X,F4.1,6X,F6.1,3X,F4.1,5X,F4.1)
149
150      WRITE (ITAPE,3000)IND,(F(1),I=1,4)
151      FORMAT(2X,I4,18X,F6.1,19X,F6.1,17X,F6.1,20X,F6.1)
152
153      CALL TO SUBROUTINE HEDR2 TO WRITE THE SMA MEASUREMENTS
154
155      CALL HEDR2(ITAPE)
156
157      WRITE (ITAPE,3100)BSTME,A(5),ASD(5),A(6),ASD(6),AMSP,
158      *AVSPT,ANMHT,ANMAT,ANMRT,SNDIST,AUCOR,SNANG(1)
159      FORMAT(3X,A8,1X,'C',1X,F6.1,1X,F4.1,1X,'C',1X,F6.1,1X,F4.1,
160      *2X,F4.1,2X,F4.1,2X,F4.1,1X,F4.1,1X,F4.1,4X,F7.5,
161      *9X,F7.5,14X,F4.2,1X,'EARTH')
162
163      WRITE (ITAPE,3200)B(5),BSD(5),B(6),BSD(6),BMSPT,8MWSPT,8NMHT,BNMAT,
164      *BNMRT,SNANG(2)
165      FORMAT(12X,'D',1X,F6.1,1X,F4.1,1X,'O',1X,F6.1,1X,F4.1,2X,F4.1,
166      *2X,F4.1,2X,F4.1,1X,F4.1,1X,F4.1,41X,F4.2,1X,'SOLAR')
167
168      WRITE (ITAPE,3300)F(5)
169      FORMAT(28X,F6.1)
170      IF(IREC.GT.2)THEN
171      IREC=0

```

```

1 167 ENDDIF
168 GO TO 1
169 WRITE(6,4000)IOS
170 99 FORMAT(' ERROR ENCOUNTERED= ',I6)
171 4000 RETURN
172 4010 FND

```

---VARIABLE MAP---(L0=A)	---BLOCK---	---PROPERTIES---	---TYPE---	---SIZE---	---NAME---	---ADDRESS---	---BLOCK---	---PROPERTIES---	---TYPE---	CHAR*8
A	761B		REAL	6	B TIME	1052B				REAL
ADATE	1050B		CHAR*8		BWSPT	7B	/TVAR/			REAL
AMSPT	2B	/TVAR/	REAL		BWSSP	1067B				REAL
AMSSP	1060B		REAL		BWTSP	5B	/TVAR/			REAL
AMTSP	0B	/TVAR/	REAL		C	775B				REAL
ANMAT	1063B		REAL		D	1003B				REAL
ANMBT	1064B		REAL		E	1011B				REAL
ANMHT	1062B		REAL		F	1017B				REAL
ANS	NONE	UNUSED/*S*	CHAR*1		G	1076B				REAL
AREA	6B	/COEFF/	REAL	5	I	1057B				INTEGER
ASD	1024B		REAL	6	IDAY	1065B				INTEGER
ATIME	1051B		CHAR*8		IDF	1	DUMMY-ARG			INTEGER
AUCOR	1100B		REAL		INO	1056B				INTEGER
AWSP	6B	/TVAR/	REAL		IOS	1055B				INTEGER
AWSSP	1061B		REAL		IREC	1054B				INTEGER
AWTSP	4B	/TVAR/	REAL		ITAPE	1073B				INTEGER
R	767B		REAL	6	JDAY	1074B				INTEGER
RMSPT	3B	/TVAR/	REAL		N	1075B				INTEGER
RMSSP	1066B		REAL		PRODCT	1042B				REAL
RMTSP	1R	/TVAR/	REAL		RESTN	0B	/COEFF/			REAL
RNMAT	1071B		REAL		SNANG	1040B				REAL
RNMBT	1072B		REAL		SNDIST	1077B				REAL
RNMHT	1070B		REAL		SPMOD	1047B				CHAR*7
RSD	1032B		REAL	6	SWTRN	13B	/COEFF/			REAL
RSTME	1053B		CHAR*8							

ORIGINAL PAGE IS  
OF POOR QUALITY

ORIGINAL PAGE IS  
 OF POOR QUALITY

```

SUBROUTINE NOA9(F,SPMOD,IOF,ITAPE)
  C
  C
  C
  COMMON /TVAR/AMTSP,BMTSP,AMSPT,BMSPT,AWTSP,BWTSP,AMSPT,BMSPT
  CHARACTER SPMOD*7
  DIMENSION F(5)
  PPFNW = 0.23
  PPSNW = 4.76
  PPSNM = 1.35
  PPFNM = 1.15
  IF(IDF.EQ.3)THEN
    SPMOD = 'NOAA-10'
    ITAPE=7
  ELSEIF(IDF.EQ.1)THEN
    SPMOD = 'NOAA-9'
    ITAPE = 4
  ENDIF
  PRINT 125,ITAPE
  FORMAT(I3)
  F(1) = F(1)+(AMTSP-BMTSP)*PPFNW
  F(1) = F(1)+(AMSPT-BMSPT)*PPSNM
  F(3) = F(3)+(AWTSP-BWTSP)*PPFNW
  F(3) = F(3)+(AMSPT-BMSPT)*PPSNW
  PRINT 118,F(1),F(3)
  FORMAT(F7.2,F7.2)
  RETURN
  END
  
```

LINE	NAME	ADDRESS	BLOCK	PROPERTIES	SIZE	TYPE
1	AMSPT	28	/TVAR/			REAL
2	AMTSP	08	/TVAR/			REAL
3	AMSPT	68	/TVAR/			REAL
4	AMTSP	48	/TVAR/			REAL
5	BMSPT	38	/TVAR/			REAL
6	BMTSP	18	/TVAR/			REAL
7	BMSPT	78	/TVAR/			REAL
8	BWTSP	58	/TVAR/			REAL
9	F	1	DUMMY-ARG			REAL
10	IDF	3	DUMMY-ARG			INTEGER
11	ITAPE	4	DUMMY-ARG			INTEGER
12	PPFNW	1308				REAL
13	PPSNW	1258				REAL
14	PPSNM	1278				REAL

```

1 SUBROUTINE ERBS(F,SPMOD,ITAPE)
2 COMMON/TVAR/AMTSP,BMTSP,AMSPT,BMSPT,AWTSP,BWTSP,AMSPT,BWSPT
3 CHARACTER SPMOD*7
4 DIMENSION F(5)
5 PPFEW = 0.21
6 PPSEW = 5.04
7 PPFEM = 0.88
8 PPSM = 2.45
9 SPMOD = 'ERBS '
10 ITAPE = 5
11 F(1) = F(1) + (AMTSP-BMTSP)*PPFEM
12 F(1) = F(1) + (AMSPT-BMSPT)*PPSEW
13 F(3) = F(3) + (AWTSP-BWTSP)*PPFEW
14 F(3) = F(3) + (AMSPT-BMSPT)*PPSEM
15 PRINT 118,F(1),F(3)
16 FORMAT(F7.2,2X,F7.2)
17 RETURN
18 END
  
```

NAME	ADDRESS	BLOCK	PROPERTIES	SIZE	TYPE	PROPERTY
AMSPT	2B	/TVAR/			REAL	
AMTSP	0B	/TVAR/			REAL	
AWSPT	6B	/TVAR/			REAL	
AWTSP	4B	/TVAR/			REAL	
BMSPT	3B	/TVAR/			REAL	
BMTSP	1B	/TVAR/			REAL	
BWSPT	7B	/TVAR/			REAL	
BWTSP	5B	/TVAR/			REAL	
F						
ITAPE	1	DUMMY-ARG			REAL	
PPFEM	3	DUMMY-ARG			INTEGER	
PPFEW	76B				REAL	
PPFEM	74B				REAL	
PPSEM	77B				REAL	
SPMOD	75B				REAL	
SPMOD	2	DUMMY-ARG			CHAR*7	

ORIGINAL PAGE IS OF POOR QUALITY



ORIGINAL PAGE IS  
 OF POOR QUALITY.

```

1  SUBROUTINE HEADR(ITAPE,SPMOD,PRODDT,IREC)
2
3  C
4  C
5  C
6  C
7  C
8  COMMON/COEFF/RESTN(6),AREA(5),SWTRN(2)
9  DIMENSION PRODDT(5)
10 IREC=IREC+1
11 IF(IPEC.EQ.1 .OR. IREC .GT.3)THEN
12 WRITE(ITAPE,3)
13 FORMAT(1H1,49X,1ERBE NS SOLAR CALIBRATIONS')
14 WRITE(ITAPE,5)
15 FORMAT (2X,'INPUT DATA')
16 WRITE (ITAPE,10)
17 FORMAT (20X,'MED-TOT',14X,'MED-SW',12X,'WIDE-TOT',11X,'WIDE-SW',
18 *12X,'SHA')
19 WRITE(ITAPE,15)(AREA(I),I=1,5)
20 FORMAT(5X,'AREA',11X,4(F10.8,9X),F10.8)
21 WRITE(ITAPE,20)(SWTRN(I),I=1,2)
22 FORMAT(5X,'TRANSMISSION',3X,1.0,16X,F5.3,14X,1.0,16X,F5.3)
23 WRITE (ITAPE,25)(RESTN(I),I=1,4),RESTN(6)
24 FORMAT(5X,'RESISTANCE',2X,F6.1,13X,F6.1,13X,F6.1,13X,
25 *F6.1,13X,F6.1)
26 WRITE(ITAPE,30)(PRODDT(I),I=1,5)
27 FORMAT(5X,'PPODDT',7X,F7.4,12X,F7.4,12X,F7.4,12X,F7.4,12X,F7.4)
28 ENDIF
29 WRITE (ITAPE,1000)SPMOD
30 FORMAT('0',1X,A7)
31 WRITE (ITAPE,2000)
32 FORMAT(2X,'DATE',6X,'TIME',7X,'MEDIUM TOTAL CHANNEL',7X,
33 *'MEDIUM SW CHANNEL',7X,'WIDE TOTAL CHANNEL',7X,'WIDE SW CHANNEL')
34 WRITE (ITAPE,3000)
35 FORMAT(24X,'COUNTS',3X,'SD',4X,'NMFTLT',4X,'COUNTS',3X,'SD',
36 *4X,'NMFSLT',2X,'COUNTS',3X,'SD',4X,'NMFTLT',5X,'COUNTS',4X,'SD',
37 *5X,'NMFSLT')
38 RETURN
39 END

```

74/R60 OPT=1,ROUND= A/ S/ M/-D,-DS FTN 5.1+642 87/05/20. 15.09.32  
DO=--LONG/-OT,ARG= COMMON/-FIXED,CS= USER/-FIXED,DB=TB/-SB/-SL/-ER/-ID/-PMD/-ST,-AL,PL=5000  
FTN5,I=SOLCA,L=LF.

```

1 SUBROUTINE HEDR2(ITAPE)
2
3 C
4 C
5 C
6
7 WRITE(ITAPE,1000)
8 FORMAT(2X,'TIME',8X,'SOLAR MONITOR CHANNEL')
9 WRITE (ITAPE,2000)
10 FORMAT(14X,'SPACE',3X,'SD',4X,'SUN',5X,'SD',3X,'MSPT',2X,
11 *'WSPT',4X,'SMA TEMPS',5X,'SUN DISTANCE',4X,
12 *'1 AU CORRECTION',7X,'SUN ANGLE')
13 RETURN
14 END

```

THIS ROUTINE PRINTS THE SOLAR MONITOR CHANNEL HEADRS

--VARIABLE MAP---(LO=A)  
--NAME---ADDRESS ---BLOCK-----PROPERTIES-----TYPE-----SIZE

ITAPE 1 DUMMY-ARG INTEGER

--STATEMENT LABELS---(LO=A)  
--LABEL-ADDRESS-----PROPERTIES-----DEF

1000 13B FORMAT 7  
2000 20B FORMAT 9

--ENTRY POINTS---(LO=A)  
--NAME---ADDRESS---ARGS---

HEDR2 3B 1

--STATISTICS--

PROGRAM-UNIT LENGTH 43B = 35  
CM STORAGE USED 61300B = 252A0  
COMPILE TIME 0.144 SECONDS

Standard Bibliographic Page

1. Report No. NASA CR-178350		2. Government Accession No.		3. Recipient's Catalog No.	
4. Title and Subtitle Data Analysis and Software Support for the Earth Radiation Budget Experiment				5. Report Date August 1987	
				6. Performing Organization Code	
7. Author(s) W. Edmonds and S. Natarajan				8. Performing Organization Report No.	
				10. Work Unit No. 665-45-30-01	
9. Performing Organization Name and Address ST Systems Corporation (STX) 28 Research Drive Hampton, VA 23666				11. Contract or Grant No. NAS1-17851	
				13. Type of Report and Period Covered Contractor Report	
12. Sponsoring Agency Name and Address NASA Langley Research Center Hampton, VA 23665				14. Sponsoring Agency Code	
15. Supplementary Notes  Langley Technical Monitor: D. E. Hinton					
16. Abstract  Computer programming and data analysis efforts were carried out under this contract in support of the Earth Radiation Budget Experiment (ERBE) at NASA/Langley. In this final report there will be a brief description of ERBE followed by sections describing software development and data analysis for both pre-launch and post-launch instrument data.					
17. Key Words (Suggested by Authors(s))  ERBE ERBS NOAA-9 NOAA-10			18. Distribution Statement  Unclassified - Unlimited  Subject Category 61		
19. Security Classif.(of this report) Unclassified		20. Security Classif.(of this page) Unclassified		21. No. of Pages 91	22. Price A05

For sale by the National Technical Information Service, Springfield, Virginia 22161