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
Unclas
G3/61 0087061

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

TABLE OF CONTENTS

	Page
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.

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^{o}. NOAA-9 and NOAA-10 were placed into polar orbit at 99^{o} 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

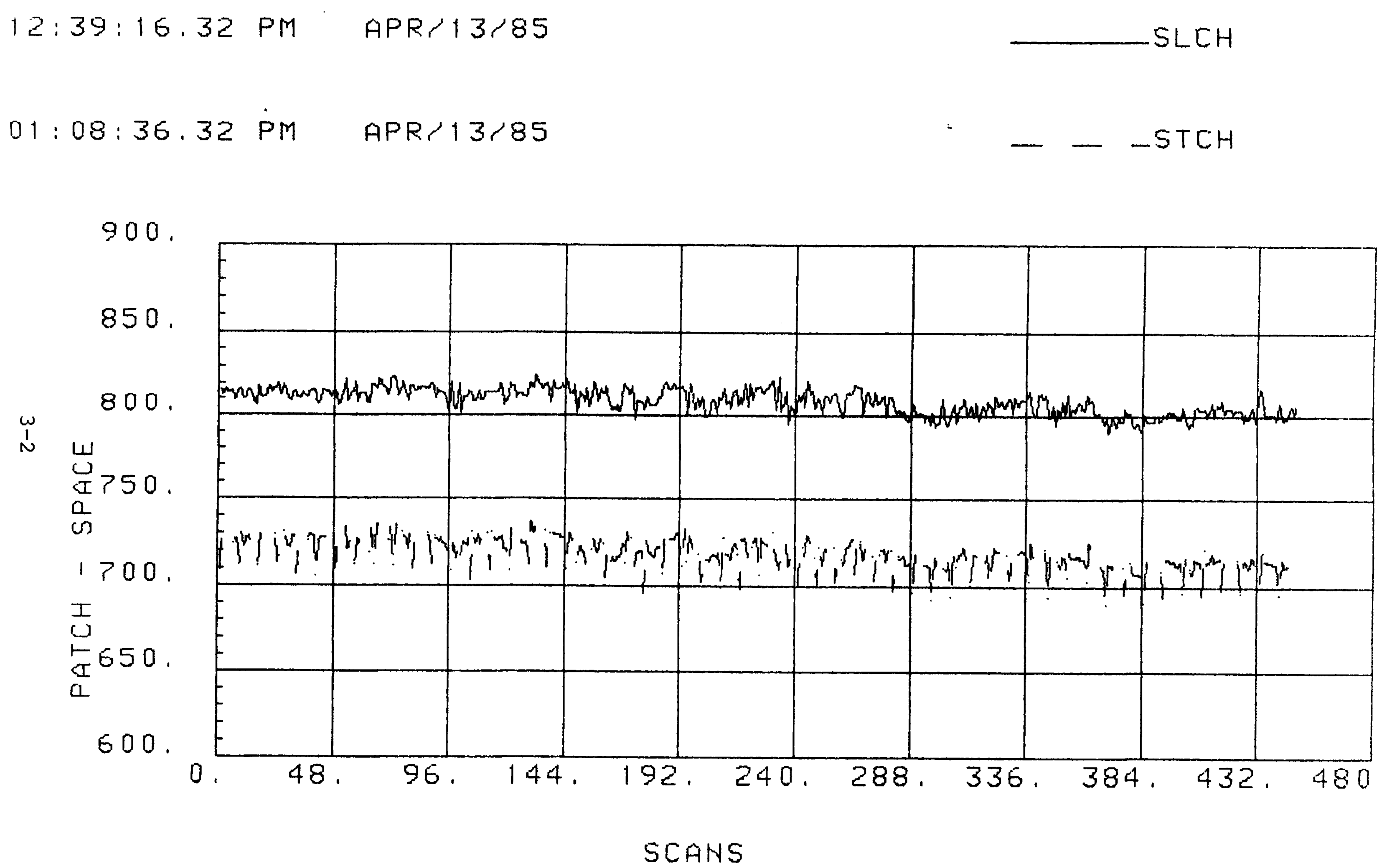
12:39:16.32 PM APR/13/85
01:08:36.32 PM APR/13/85
SLCH
STCH
900.
850.
800.
750.
700.
650.
600.
PATCH - SPACE
0.
48.
96.
144.
192.
240.
288.
336.
384.
432.
480
SCANS

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

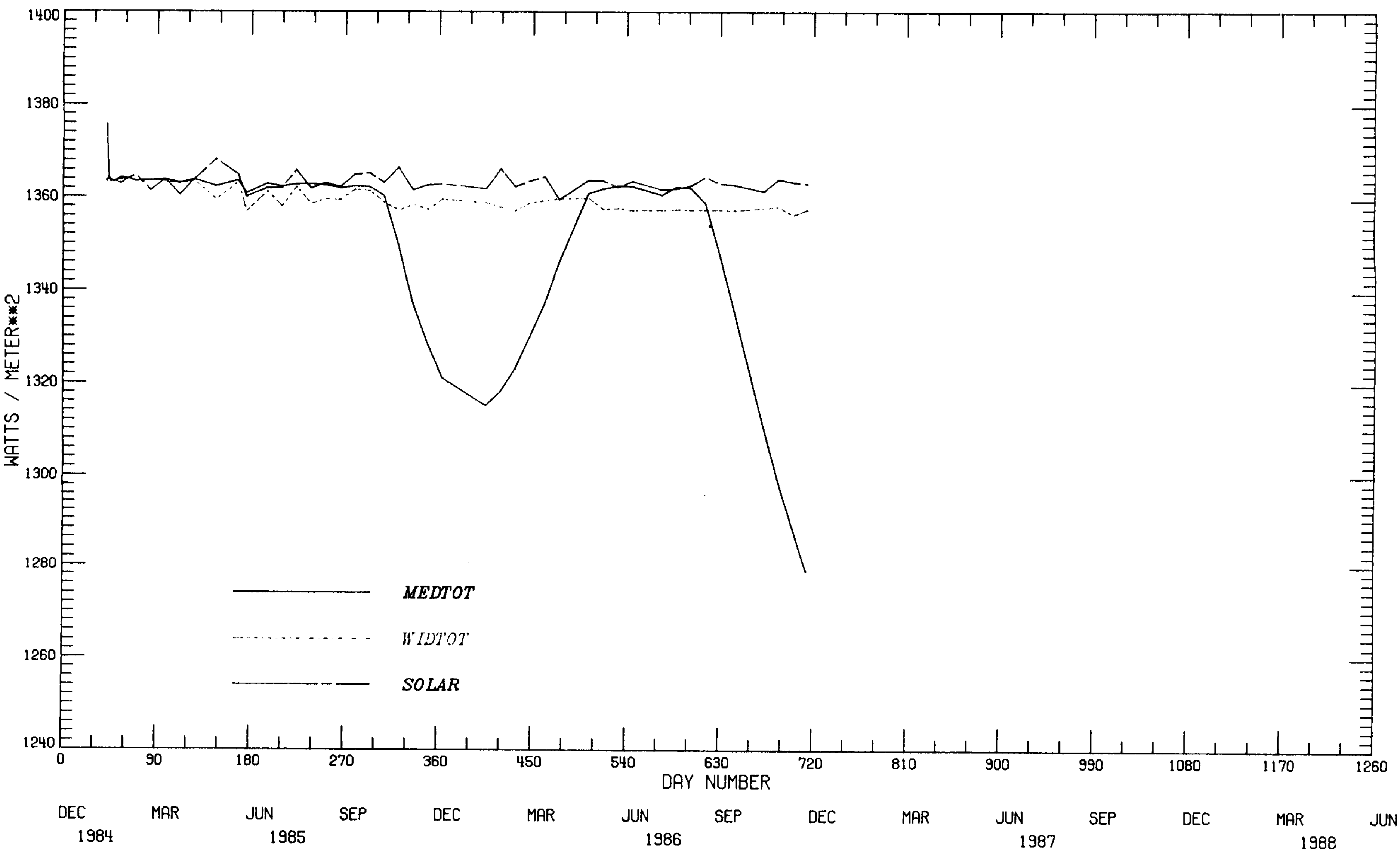

Figure 2

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

```
PROGRAM SOCC; { DISPLAY PROGRAM FOR ERBE/NOAA-9 & NOAA-10 }
{

written by :    William L. Edmonds, STX Corporation
                December 1984

modified   :    Summer 1986 for NOAA-10 capability.

Program SOCC inputs ERBE/TIROS data from the ADCCP BUFFER, processes
and limitchecks 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.    }



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

TYPE integBUFF = ARRAY[0..4000] OF INTEGER ;
TYPE BITE = ARRAY[0..80] OF BYTE ;
TYPE MinorFrameBuffs = Array[0..103] of byte ;
Type dualbuffs = Array[0..4] of MinorFrameBuffs  ;
TYPE MinorFramewords = Array[0..51] OF INTEGER ;
Type dualwordbuffs = Array[0..4] of MinorFRamewords ;
TYPE  ERBREC = ARRAY[0..21] OF BYTE ;
     { USED IN SAVING ERBE DATA TO DISK }

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

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

     ON : STRING[3] = ' ON' ;{USED FOR DIGB DATA}
     OFF : STRING[3] = 'OFF' ;{USED FOR DIGB DATA}
     NOAA9 : STRING[10] = 'NOAA9.DAT ' ;
     NOAA10 : STRING[10] = 'NOAA10.DAT' ;
     WHITE: INTEGER = 15 ;  { color display attribute values }
     RED: INTEGER = 12 ;
     YELLOW : INTEGER = 14 ;
     GREEN : INTEGER = 10 ;
```

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

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

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

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

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

      VALST : ABC ;
```

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

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

    RCVBUFW : COMBUFFW ABSOLUTE $8F80:$0004 ;
    msgbuf : smallbufw absolute $8F80: $0000 ;
    XMTBUF : COMBUFF ABSOLUTE $8F80: $0400 ;
    MinorFrame : dualbuffs absolute $7f80 : $0004 ; { 9800:0004 ;  }
    MinorFrameW : dualwordbuffs absolute $7f80 : $0004 ; { 9800:0004 ; }
    MINFW : COMBUFFW ABSOLUTE $7f80 : $0004  ; { 9800:0004 ;}
  FRAME: integBUFF ABSOLUTE $B800:$0000;
    BFRAME: BYTEBUFF ABSOLUTE $B800:$0000;
    txt : text ;
    txtfile : string[10] ;
    SORC:ABC ; {TEXT STRING USED FOR CONOGRAPHICS DISPLAY}
    ATT, XY: INTEGER;   I,K,   J:INTEGER;
    SCRNINT: ARRAY[0..1] OF INTEGER ABSOLUTE $0000:$0014;
    STORINT : ARRAY[0..1] OF INTEGER ;
    statpr : byte absolute $0050:$0000;
        {INTERRUPT VECTOR LOCATION FOR PRINTING}



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

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



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

```
  At termination, it returns the screen to 25 X 80 . }

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

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

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

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

function GETKEY : integer ; {GET VALUE OF KEY STRUCK...
                             IF NO KEY THEN ZERO IS RETURNED }
begin
     with recpack do    {SET UP FOR INTERRUPT 21 HEX WITH AH = 6
                         AND DX = FF HEX }
          begin
               ah := 6 ;
               al := 0;
               ax := ah shl 8 + al ;
               dx := $ff ;
          end;
          intr($21,recpack);    { DO INTERRUPT 21 }
          with recpack do
          begin
               al := ax and $ff ;   { GET VALUE OF CHARACTER }
          end;
          GETKEY := al ;   { IF NO KEY WAS HIT, ZERO WILL BE RETURNED}
end;



function CHKTIM : boolean ;
```

```
              { TIMING ROUTINE FOR ADCCP COMMUNICATIONS}
BEGIN
     AH := $2C; { SET UP FOR INTERRUPT 21 HEX WITH AH = 2C HEX}
     WITH RECPACK DO
     BEGIN
          AX := AH SHL 8 ;
     END;
     INTR($21,RECPACK);
     WITH RECPACK DO { EXTRACT THE TIME }
     BEGIN
          DH := DX SHR 8 ;
          IF ( dh < elpsec ) then
          begin
               CL := CX AND 255 ;
               cl := cl -1 ;
               dh := dh +60 ;
          end;
          dh := dh - elpsec ;
          if ( dh < maxsec ) then
          begin
               if ( cl <> elpmin ) then chktim := true
               else chktim := false ;
          end
          else chktim := true ;
     END;

END;

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

VAR I : INTEGER ;
BEGIN
     CASE ID OF

     0:   FOR I := 0 TO 3 DO
          XMTBUF[I] := ID0MSG[I] ;

     1:  FOR I := 0 TO 3 DO
         XMTBUF[I] := ID1MSG[I] ;
     else i:= i ;
     END; { OF CASE }
     PORT[BISYNC] := XMTRCV ;
     AH := $2C;
     WITH RECPACK DO
     BEGIN
          AX := AH SHL 8 ;
     END;
     INTR($21,RECPACK);
     WITH RECPACK DO
     BEGIN
```

```
                DH := DX SHR 8 ;
                CL := CX AND 255 ;
                ELPSEC := DH ;
                ELPMIN := CL ;
           END;
      END;  { OF SENDIDLE }

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

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

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

      BEGIN
           ID := 0 ;
      WAIT:     SENDIDLE ;
                STATUS := PORT[BISYNC] AND MSGRCV ;
                IF STATUS <> 0 THEN GOTO MESSRCV ;
                IF NOT(CHKTIM) THEN GOTO WAIT ;
                STATUS := TIMERR ;
                GOTO XIT ;

      MESSRCV:   IF RCVBUF[0] <> 0 THEN
                BEGIN
                     STATUS := CSERR ;
                     GOTO XIT ;
                END
                ELSE
                BEGIN
                     MSGID := RCVBUF[3] AND 4 ;
                     IF MSGID <> 0 THEN
                     BEGIN
                          STATUS := PTLERR ;
                          GOTO XIT ;
                     END;
```

```
          END;
          ID := 1;
          SENDIDLE ;
          STATE := 1;
          STATUS := NOERR ;

XIT: END;


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

AGIN: STATUS := PORT[BISYNC] AND MSGRCV ;
      IF STATUS <> 0 THEN GOTO GOTMSG ;
      IF (NOT(CHKTIM)) THEN GOTO AGIN ;
      SENDIDLE;
      STATUS := TIMERR;
      GOTO DONE;
GOTMSG: IF (RCVBUF[0] <> 0 )THEN
        BEGIN
             SENDIDLE;
             STATUS := CSERR;
             GOTO DONE ;
        END;
        MSGID :=  (RCVBUF[3] AND 4) SHR 2 ;
        IF (MSGID = STATE) THEN goto gdmsg ;
        if ( state = 1 ) then goto gdms00 ;
        BEGIN
             SENDIDLE ;
             STATUS := PTLERR;
             GOTO DONE;
        END;
gdmsg:        STATE := (NOT(STATE)AND 1) ;
        ID := STATE ;
gdms00:        XFER;
        SENDIDLE;
        STATUS := NOERR;

DONE: END;

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


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

```
         SCRNMODE[0] := $71;
         SCRNMODE[1] := $50;
         SCRNMODE[2] := $5A;
         SCRNMODE[3] := $0F;
         SCRNMODE[4] := $1B;
         SCRNMODE[5] := 6;
         SCRNMODE[6] := $19;
         SCRNMODE[7] := $1A;
         SCRNMODE[8] := 3;
         SCRNMODE[9] := 7;
         SCRNMODE[10] := $20 ;
         SCRNMODE[11] := $20 ;
         SCRNMODE[12] := 0;
         SCRNMODE[13] := 0;
         SCRNMODE[14] := 0;
         SCRNMODE[15] := 0;
         S5080(SCRNMODE[0]); { CALL ROUTINE TO SEND DATA TO CONOGRAPHICS }

    END;

    PROCEDURE OLDSCREEN ; { RESTORES SCREEN TO NORMAL MODE }
    VAR LOC : INTEGER ;
    BEGIN
          FOR LOC := 0 TO 3999 DO
          FRAME[LOC] := $F00 ;

          SCRNMODE[4] := $1F ;
          SCRNMODE[7] := $1C ;
          SCRNMODE[8] := 2;
          SCRNMODE[10] := 6;
          SCRNMODE[11] := 7;
          S5080(SCRNMODE[0]);
    END;


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

    BEGIN
         ATT := GREEN ;
         IF X < YL THEN
         BEGIN
              IF X < RL THEN
              BEGIN
                   ATT := RED ;
                   SORC := SORC + 'RL' ;
              END
              ELSE
              BEGIN
```

```
               ATT := YELLOW ;
               SORC := SORC + 'YL' ;
          END;
     END
     ELSE
     BEGIN
     IF X > YH THEN
     BEGIN
          IF X > RH THEN
          BEGIN
               ATT := RED ;
               SORC := SORC + 'RH' ;
          END
          ELSE
          BEGIN
               ATT := YELLOW ;
               SORC := SORC + 'YH' ;
          END;
     END
     ELSE SORC := SORC + '    ';
     END;
END; {OF LIMITCHECK }


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

VAR TIMSTRING : ABC ;
DAYS,HRS,MINS : INTEGER;  millisecs,SECS : REAL ;
BEGIN
    days := (time[0]shl 1) +((time[1] and 128)shr 7) ;
    millisecs := ((( time[1]and 7)*256.0 + time[2])*256.0 + time[3])
                     *256.0 +time[4];
    hrs := trunc(millisecs/3600000.0) ;
    mins := trunc(millisecs/60000.0) mod 60 ;
    secs := trunc((millisecs/1000.0)-mins*60.0-hrs*3600.0);
    str(days:4,sorc); putout(sorc,frame[xypos(42,40)],white);
    str(hrs:2,sorc); putout(sorc,frame[xypos(42,45)],white);
    str(mins:2,sorc);  putout(sorc,frame[xypos(42,50)],white);
    str(secs:6:3,sorc); putout(sorc,frame[xypos(42,55)], white);
END;


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

VAR I: INTEGER;
BEGIN
 txtfile := 'sorc.txt' ;
 assign(txt,txtfile);
 reset(txt);
```

```
 att := 15 ;
i := 80 ;
while not eof(txt) do
begin
readln(txt,sorc);
sorc := sorc + '                              ';
putout(sorc,frame[i],att);
i := i + 80 ;
end;
close (txt);

END;


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

LABEL RETURN ;
var nextmf,tipstatus: integer ;   nmf,mf: string[4] ;
BEGIN
     { CALCULATE EXPECTED MINOR FRAME NUMBER AND CHECK THE ACTUAL
      MINOR FRAME NUMBER RECEIVED. IF NOT EQUAL, DISPLAY DIAGNOSTIC
      MESSAGE }
      nextmf := minorframenum +1 ;
      if (nextmf > 319 ) then nextmf := 0 ;
      MINORFRAMENUM := MINORFRAME[N,5] + (MINORFRAME[N,4]AND 1)shl 8;
      if (minorframenum<>nextmf) then
      begin
           str(nextmf:4,nmf);
           str(minorframenum:4,mf);
           sorc := 'expecting mf ' + nmf + ' but found mf ' + mf ;
           putout(sorc,frame[xypos(45,2)],yellow);
      end ;
      { CHECK SPACE CRAFT ID TO DETERMINE NOAA-9,NOAA-10 OR OTHER
        NOAA SATELLITE. WE ARE INTERESTED ONLY IN NOAA-9 & 10 }
      SCID := MINORFRAME[N,2] AND 15 ;
      IF ((SCID <> $0D)AND(SCID<>$0F)AND(SCID<>$00)
      AND(SCID<>$0E)) THEN GOTO RETURN ;
      {
          IF SATELLITE ID IS NEITHER NOAA-F NOR NOAA-G
          }
      { NOAA-9 CAN HAVE EITHER A HEX '0D' OR '0E' ID }
      { NOAA-10 CAN HAVE EITHER A HEX '0F' OR '00' ID }
      IF (( SCID = $0D) OR(SCID = $0E)) THEN FORG := ' 9'
       ELSE
      FORG := '10' ;
     { IF THE ID HAS CHANGED SINCE THE LAST FRAME OF DATA,
     CLEAR THE SCREEN AND START WITH A NEW TEMPLATE }
     IF (SCID <> OLDSCID ) THEN DISPLAYACRO ;
     OLDSCID := SCID ;
     sorc := forg ;
```

```
        putout(sorc,frame[xypos(18,40)],green);
         if (minorframenum = 0 ) then
         begin
              sorc := '                                   ';
              putout(sorc,frame[xypos(45,2)],yellow);
             FOR I := 0 TO 4 DO
             TIME[I] := MINORFRAME[N,I+8] ;
             displaytime;
         END;
         TIPSTATUS := (MINORFRAME[N,3] AND 96) SHR 5 ;
         ATT := RED ; { DEFAULT ATTRIBUTE COLOR, ONLY ORBIT MODE IS GREEN }
         CASE TIPSTATUS OF

   0:    BEGIN
              SORC := 'ORBIT MODE     ' ;
              ATT := GREEN ;
         END;
   1:    SORC := 'DUMP MODE      ' ;
   2:    SORC := 'DWELL MODE     ' ;
   3:    SORC := 'UNDEFINED MODE' ;
   else sorc := 'undefined mode' ;
         END;
         PUTOUT(SORC,FRAME[XYPOS(41,2)],ATT);
   IF (MINORFRAMENUM > 319 ) THEN GOTO RETURN ;
         DIGB := MINORFRAME[N,12];
         ANALOG := MINORFRAME[N,13] ;
         NDIGA := (MINORFRAME[N,52] shl 8) or minorframe[n,53] ;
         SDIGA[0] := (MINORFRAME[N,18] shl 8 or minorframe[n,19] )SHR 4 ;
         SDIGA[1] :=(((MINORFRAME[N,19] AND 15) SHL 8) OR MINORFRAME[N,28]);
         SDIGA[2] := (MINORFRAME[N,29] SHL 4) OR ( MINORFRAME[N,44] SHR 4) ;
         SDIGA[3] := ((MINORFRAME[N,44] and 15)shl 8 or minorframe[n,45] ) ;
         SDIGA[4] := (MINORFRAME[N,60] shl 8 or minorframe[n,61] )SHR 4;
         SDIGA[5] :=(((MINORFRAME[N,61] AND 15 ) SHL 8) OR MINORFRAME[N,72]);
         SDIGA[6] := (MINORFRAME[N,73] SHL 4) OR (MINORFRAME[N,86] SHR 4 ) ;
         SDIGA[7] :=(( MINORFRAME[N,86] and 15) shl 8 or minorframe[n,87] );
   { ERBDAT ARRAY IS USED TO SAVE THE RAW VALUES OF THE ERBE DATA FROM THE
     NOAA 'TIP' DATA STREAM. OUT OF EACH 520 BYTE BUFFER RECEIVED FROM
     THE ADCCP BOX, 5 MINOR FRAMES OF DATA WITH 22 BYTES OF USEFUL INFO
     EACH IS SAVED TO DISK }
         ERBDAT[0] := (MINORFRAME[N,4] AND 1) OR (SCID SHL 1) ;
         ERBDAT[1] := MINORFRAME[N,5] ;
         ERBDAT[2] := MINORFRAME[N,8];
         ERBDAT[3] := MINORFRAME[N,9];
         ERBDAT[4] := MINORFRAME[N,10] ;
         ERBDAT[5] := MINORFRAME[N,11] ;
         ERBDAT[6] := MINORFRAME[N,12] ;
         ERBDAT[7] := MINORFRAME[N,13] ;
         ERBDAT[8] := MINORFRAME[N,18] ;
         ERBDAT[9] := MINORFRAME[N,19] ;
         ERBDAT[10] := MINORFRAME[N,28] ;
         ERBDAT[11] := MINORFRAME[N,29] ;
```

```
         ERBDAT[12] := MINORFRAME[N,44] ;
         ERBDAT[13] := MINORFRAME[N,45] ;
         ERBDAT[14] := MINORFRAME[N,52] ;
         ERBDAT[15] := MINORFRAME[N,53] ;
         ERBDAT[16] := MINORFRAME[N,60] ;
         ERBDAT[17] := MINORFRAME[N,61] ;
         ERBDAT[18] := MINORFRAME[N,72] ;
         ERBDAT[19] := MINORFRAME[N,73] ;
         ERBDAT[20] := MINORFRAME[N,86] ;
         ERBDAT[21] := MINORFRAME[N,87] ;
   IF (FORG = ' 9') THEN  WRITE(F9,ERBDAT)
   ELSE WRITE(F10,ERBDAT) ;
         ssch := sdiga[0] ;
         slch := sdiga[1] ;
         stch := sdiga[2] ;
         scpos := sdiga[3] ;
   RETURN:
   END;  { OF DIVY }


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

   BEGIN
        CASE MINORFRAMENUM MOD 32 OF

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

               IF MINORFRAMENUM>0 THEN
               BEGIN
                   SORC := ONOFF(DIGB,16) ;
                   PUTOUT(SORC,FRAME[XYPOS(11,14)],GREEN);
              END;
            END;
        3: BEGIN         { SCANNER PULSE LOAD BUS A }
            SORC := ONOFF(DIGB,64);
            IF(SORC = ON ) THEN ESIPL := '   A' ELSE IF (ESIPL = '   A')
            THEN   ESIPL := OFF ;
            PUTOUT(ESIPL,FRAME[XYPOS(12,54)],GREEN);

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

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

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

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

            { NON-SCANNER SPARE WOULD GO HERE ALSO }
            END;
         8: BEGIN          { SCANNER PED STANDBY HEATER }
            SORC := ONOFF(DIGB,16);
            PUTOUT(SORC,FRAME[XYPOS(14,54)],GREEN);
            END;
        10: BEGIN          { NON-SCANNER INSTRUMENT HEATER PWR }
             SORC := ONOFF(DIGB,64);
             PUTOUT(SORC,FRAME[XYPOS(34,12)],GREEN);
             END;
        11: BEGIN             { NON-SCANNER ELEVATION MOTOR POWER }
             SORC := ONOFF(DIGB,128);
             PUTOUT(SORC,FRAME[XYPOS(31,34)],GREEN);
             END;
        13: BEGIN              { SCANNER PULSE LOAD BUS B }
             SORC := ONOFF(DIGB,64);
             IF(SORC = ON )
             THEN ESIPL := '  B'
             ELSE IF (ESIPL = '  B' )
                  THEN
                  ESIPL := OFF ;
             PUTOUT(ESIPL,FRAME[XYPOS(12,54)],GREEN);
             END;
        17: BEGIN              { SCANNER INSTR POWER }
             SORC := ONOFF(DIGB,128);
             PUTOUT(SORC,FRAME[XYPOS(10,54)],GREEN);

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

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

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

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

```
              PUTOUT(SORC,FRAME[XYPOS(32,12)],GREEN);
              END;
          19: BEGIN    { NON-SCANNER PED. STANDBY HEATER}
              SORC := ONOFF(DIGB,128);
              PUTOUT(SORC,FRAME[XYPOS(31,12)],GREEN);
              END;
           else i := i ;
             END;     { OF CASE }


END;

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


FUNCTION EQU2(COUNTS:INTEGER):REAL ;
BEGIN
    EQU2 := COUNTS/409.5 ;
END;

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

END;

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

END;

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

END;

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

END;

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

END;
```

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

END;

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

END;


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

END;

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

END;

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

END;

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

END;

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

END;
```

```
FUNCTION EQU14(N:REAL):REAL;
BEGIN
     EQU14 := ((((-3.7944378*N+45.022096)*N-211.33864)*N+489.0738)
              *N -583.00496)*N+347.83511;

END;

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

END;

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

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

4:   BEGIN   { SCANNER CHAN 0 ELECTRIC SLICE 2 TEMP NOT AVAILABLE }
     END;
5:   BEGIN   { NON-SCANNER CHAN 0 ELECTRIC SLICE 2 TEMP }
     RANALOG := EQU15(ANALOG/51.0)  ;
     STR(RANALOG:5:3,SORC);
     LIMITCHECK(RANALOG,-10.0,0.0,40.0,50.0,SORC,ATT);
     OUTPUT(SORC,FRAME[XYPOS(37,71)],att);
     END;
13:  BEGIN    { NON-SCANNER CHAN 1 ELECTRIC SLICE 3 TEMP }
     RANALOG := EQU15(ANALOG/51.0)  ;
     STR(RANALOG:5:3,SORC);
     LIMITCHECK(RANALOG,-10.,0.0,40.0,50.0,SORC,ATT);
     OUTPUT(SORC,FRAME[XYPOS(34,71)],GREEN);
     END;
27:  BEGIN    { SCANNER CHAN 3 POWER CONVERTER TEMP }
     RANALOG := EQU14(ANALOG/51.0);
     STR(RANALOG:5:3,SORC);
     LIMITCHECK(RANALOG,-10.0,0.0,45.0,50.0,SORC,ATT);
     OUTPUT(SORC,FRAME[XYPOS(9,71)],ATT);
     END;
28:  BEGIN    { NON-SCANNER CHAN 3 POWER CONVERTER TEMP }
     RANALOG := EQU14(ANALOG/51.0);
     STR(RANALOG:5:3,SORC);
     LIMITCHECK(RANALOG,-10.0,0.0,40.0,50.0,SORC,ATT);
     OUTPUT(SORC,FRAME[XYPOS(33,71)],ATT);
     END;
```

```
35:  BEGIN     {SCANNER CHAN 4 BOX BEAM TEMP }
     RANALOG := EQU14(ANALOG/51.0) ;
     STR(RANALOG:5:3,SORC);
     LIMITCHECK(RANALOG,0.0,10.0,34.0,36.0,SORC,ATT);
     OUTPUT(SORC,FRAME[XYPOS(7,71)],ATT);
     END;
43:  BEGIN      { SCANNER CHAN 5   +5 VOLT MONITOR }
     RANALOG := 2.0*ANALOG/51.0 ;
     STR(RANALOG:5:3,SORC);
     LIMITCHECK(RANALOG,3.5,4.0,6.0,6.5,SORC,ATT);
     OUTPUT(SORC,FRAME[XYPOS(2,71)],ATT);
     END;
45:  BEGIN      { NON-SCANNER CHAN 5  +5 VOLT MONITOR }
     RANALOG := 2.0*ANALOG/51.0 ;
     STR(RANALOG:5:3,SORC);
     LIMITCHECK(RANALOG,3.5,4.0,6.0,6.5,SORC,ATT);
     OUTPUT(SORC,FRAME[XYPOS(39,12)],ATT);
     END;
51:  BEGIN      { SCANNER CHAN 6   -15 VOLT MONITOR }
     RANALOG := -6.0*ANALOG/51.0 ;
     STR(RANALOG:5:3,SORC);


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


     OUTPUT(SORC,FRAME[XYPOS(5,71)],ATT);
     END;
53:  BEGIN      { NON-SCANNER CHAN 6  -15 VOLT MONITOR };
     RANALOG := -6.0*ANALOG/51.0 ;
     STR(RANALOG:5:3,SORC);
     LIMITCHECK(RANALOG,-16.5,-16.0,-14.0,-13.5,SORC,ATT);
     OUTPUT(SORC,FRAME[XYPOS(39,71)],ATT);
     END;
59:  BEGIN      { SCANNER CHAN 7  +15 VOLT MONITOR } ;
     RANALOG := 6.0*ANALOG/51.0 ;
     STR(RANALOG:5:3,SORC);
     LIMITCHECK(RANALOG,13.5,14.0,16.0,16.5,SORC,ATT);
     OUTPUT(SORC,FRAME[XYPOS(4,71)],ATT);
     END;
60:  BEGIN      { NON-SCANNER CHAN 7   +15 VOLT MONITOR}
     RANALOG := 6.0*ANALOG/51.0;
     STR(RANALOG:5:3,SORC);
     LIMITCHECK(RANALOG,13.5,14.0,16.0,16.5,SORC,ATT);
     OUTPUT(SORC,FRAME[XYPOS(39,52)],ATT);
     END;
67:  BEGIN      { SCANNER CHAN 8  +10 VOLT MONITOR }
     RANALOG := 4.0*ANALOG/51.0 ;
     STR(RANALOG:5:3,SORC);
```

```
        LIMITCHECK(RANALOG,8.5,9.0,11.0,11.5,SORC,ATT);
        OUTPUT(SORC,FRAME[XYPOS(3,71)],ATT);
        END;
  68:   BEGIN       { NON-SCANNER CHAN 8 +10 VOLT MONITOR}
        RANALOG := 4.0*ANALOG/51.0 ;
        STR(RANALOG:5:3,SORC);
        LIMITCHECK(RANALOG,8.5,9.0,11.0,11.5,SORC,ATT);
        OUTPUT(SORC,FRAME[XYPOS(39,32)],ATT);
        END;
  99:   BEGIN       { SCANNER CHAN 12  SPACE CRAFT ADAPTER TEMP}
        RANALOG := EQU15(ANALOG/51.0);
        STR(RANALOG:5:3,SORC);
        LIMITCHECK(RANALOG,-10.0,0.0,30.0,40.0,SORC,ATT);
        OUTPUT(SORC,FRAME[XYPOS(13,71)],ATT);
        END;
  100:  BEGIN       {NON-SCANNER SPACE CRAFT ADAPTER TEMP }
        RANALOG := EQU15(ANALOG/51.0);
        STR(RANALOG:5:3,SORC);
        LIMITCHECK(RANALOG,-10.0,0.0,30.0,40.0,SORC,ATT);
        OUTPUT(SORC,FRAME[XYPOS(35,71)],ATT);
        END;
  107:  BEGIN       { SCANNER CHAN 13  PED FOOT TEMP }
        RANALOG := EQU15(ANALOG/51.0);
        STR(RANALOG:5:3,SORC);
        LIMITCHECK(RANALOG,-10.0,0.0,30.0,40.0,SORC,ATT);
        OUTPUT(SORC,FRAME[XYPOS(14,71)],ATT);
        END;
  108:  BEGIN        {NON-SCANNER CHAN 13  PED FOOT TEMP}
        RANALOG := EQU15(ANALOG/51.0) ;
        STR(RANALOG:5:3,SORC);
        LIMITCHECK(RANALOG,-15.0,-5.0,30.0,40.0,SORC,ATT);
        OUTPUT(SORC,FRAME[XYPOS(36,71)],ATT);
        END;
  116:   BEGIN    { SCANNER CHAN 1 ELECTRIC SLICE 3 TEMP }
        RANALOG := EQU14(ANALOG/51.0) ;
        STR(RANALOG:5:3,SORC);
        LIMITCHECK(RANALOG,-10.0,0.0,50.0,55.0,SORC,ATT);
        OUTPUT(SORC,FRAME[XYPOS(10,71)],ATT);
        END;
  117:  BEGIN         {NON-SCANNER CHAN 14 ELEV DRIVE TEMP }
        RANALOG := EQU14(ANALOG/51.0) ;
        STR(RANALOG:5:3,SORC);
        LIMITCHECK(RANALOG,0.0,10.0,30.0,40.0,SORC,ATT);
        OUTPUT(SORC,FRAME[XYPOS(31,71)],ATT);
        END;
  120:  BEGIN          {SCANNER CHAN 15  AZIMUTH DRIVE TEMP }
        RANALOG := EQU14(ANALOG/51.0);
        STR(RANALOG:5:3,SORC);
        LIMITCHECK(RANALOG,-25.0,-15.0,40.0,45.0,SORC,ATT);
        OUTPUT(SORC,FRAME[XYPOS(8,71)],ATT);
        END;
```

```
124: BEGIN           {NON-SCANNER CHAN 15 AZIMUTH DRIVE TEMP }
     RANALOG := EQU14(ANALOG/51.0);
     STR(RANALOG:5:3,SORC);
     LIMITCHECK(RANALOG,-15.0,-5.0,40.0,50.0,SORC,ATT);
     OUTPUT(SORC,FRAME[XYPOS(32,71)],ATT);
     END;
else i := i ;
END; {OF CASE }
END; {OF PROCESSANALOG }

FUNCTION SAREF(SAPOSL:BYTE):ABC ;
BEGIN
      CASE  (SAPOSL AND 15) OF      { CORRECTED 12/11/86 BY WLE }

4:    BEGIN
          SAREF := '0 WINDOW  ' ;
     END;

7:    BEGIN
          SAREF := '0 DEG      ' ;
      END;

8:    BEGIN
          SAREF := '90 WINDOW ' ;
      END;

11:   BEGIN
          SAREF := '90 DEG    ';
      END;

12:    BEGIN
           SAREF := '180 WINDOW';
       END;

15:    BEGIN
           SAREF := '180 DEG   ';
       END;

ELSE    SAREF :=     '   N/A    ';

END;  {OF CASE }
END;  { OF SAREF FUNCTION }


PROCEDURE ProcessScannerDIGA;
{ PROCESS DIGITAL 'B' DATA FOR BOTH SCANNER AND NONSCANNER }
BEGIN
     CASE MINORFRAMENUM MOD 40 OF

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

```
                PUTOUT(SORC,FRAME[XYPOS(2,14)],GREEN);
                STR(SLCH:5,SORC);
                PUTOUT(SORC,FRAME[XYPOS(2,34)],GREEN);
                STR(STCH:5,SORC);
                PUTOUT(SORC,FRAME[XYPOS(2,54)],GREEN);
                STR(SCPOS:5,SORC);
                PUTOUT(SORC,FRAME[XYPOS(12,14)],GREEN);

            END;
  37:       BEGIN
                SSWCSA := EQU2(SDIGA[3]);
                STR(SSWCSA:5:3,SORC);
                PUTOUT(SORC,FRAME[XYPOS(9,11)],GREEN);
            if (minorframenum<40) then { to account for multiplexing }
            begin
                IF ((SCID = $0D) or (scid = $0E)) THEN
                BEGIN      { DAC LIMITS FOR NOAA-9   }
                    SSDACV := EQU3(SDIGA[0]);
                    STR(SSDACV:5:3,SORC);
                    LIMITCHECK(SSDACV,-0.7,-0.2,0.8,1.3,SORC,ATT);
                    PUTOUT(SORC,FRAME[XYPOS(3,11)],ATT);

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

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

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

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

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

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

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

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

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

       38:          BEGIN
                    if (minorframenum <40) then
              begin
                    IF ((SCID = $0D) OR (SCID = $0E)) THEN
                    BEGIN
                        SDPBV := EQU4(SDIGA[0]);
                        STR(SDPBV:5:3,SORC);
                        LIMITCHECK(SDPBV,-200.0,85.,87.,200.,SORC,ATT);
                        PUTOUT(SORC,FRAME[XYPOS(10,31)],ATT);
                        SDNBV := EQU4(SDIGA[1]);
                        STR(SDNBV:5:3,SORC);
                        LIMITCHECK(SDNBV,-200.,-87.,-85.,200.,SORC,ATT);
                        PUTOUT(SORC,FRAME[XYPOS(11,31)],ATT);
                        STRV1 := EQU5(SDIGA[2]);
                        STR(STRV1:5:3,SORC);
                        LIMITCHECK(STRV1,-200.,-6.8,-6.4,200.,SORC,ATT);
                        PUTOUT(SORC,FRAME[XYPOS(7,31)],ATT);
                        STRV2 := EQU6(SDIGA[3]);

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

                    END
                    ELSE
                    BEGIN
                        SDPBV := EQU4(SDIGA[0]);
                        STR(SDPBV:5:3,SORC);
                        LIMITCHECK(SDPBV,-200.0,83.0,85.0,200.0,SORC,ATT);
```

```
                    PUTOUT(SORC,FRAME[XYPOS(10,31)],ATT);
                    SDNBV := EQU4(SDIGA[1]);
                    STR(SDNBV:5:3,SORC);
                    LIMITCHECK(SDNBV,-200.0,-86.5,-84.5,200.0,SORC,ATT);
                    PUTOUT(SORC,FRAME[XYPOS(11,31)],ATT);
                    STRV1 := EQU5(SDIGA[2]);
                    STR(STRV1:5:3,SORC);
                    LIMITCHECK(STRV1,-200.0,-7.45,-6.90,200.0,SORC,ATT);
                    PUTOUT(SORC,FRAME[XYPOS(7,31)],ATT);
                    STRV2 := EQU6(SDIGA[3]);

                    STR(STRV2:5:3,SORC);
                    LIMITCHECK(STRV2,-200.0,-7.05,-6.50,200.0,SORC,ATT);
                    PUTOUT(SORC,FRAME[XYPOS(8,31)],ATT);
                END;
                    SLBBT := EQU8(SDIGA[6]*STRV2/(-6.4));
                    STR(SLBBT:5:3,SORC);
                    LIMITCHECK(SLBBT,-25.0,-15.0,45.0,55.0,SORC,ATT);
                    PUTOUT(SORC,FRAME[XYPOS(5,31)],ATT);
                    STBBT := EQU8(SDIGA[7]*STRV2/(-6.4));
                    STR(STBBT:5:3,SORC);
                    LIMITCHECK(STBBT,-25.0,-15.0,45.0,55.0,SORC,ATT);
                    PUTOUT(SORC,FRAME[XYPOS(7,51)],ATT);
                    SSWCST := EQU9(SDIGA[5]);
                    STR(SSWCST:5:3,SORC);
                    LIMITCHECK(SSWCST,-25.0,-15.0,35.0,45.0,SORC,ATT);
                    PUTOUT(SORC,FRAME[XYPOS(8,11)],ATT);
                end;
                END;
    39:         BEGIN
                    if(minorframenum<40) then
                    begin
                        SSDETT := EQU7(SDIGA[1]) ;
                        STR(SSDETT:5:3,SORC);
                        LIMITCHECK(SSDETT,37.0,37.5,38.5,39.0,SORC,ATT);
                        PUTOUT(SORC,FRAME[XYPOS(4,11)],ATT);
                    end;
                    if(minorframenum=79) then
                    begin
                        SLDETT := EQU7(SDIGA[2]);
                        STR(SLDETT:5:3,SORC);
                        LIMITCHECK(SLDETT,37.0,37.5,38.5,39.0,SORC,ATT);
                        PUTOUT(SORC,FRAME[XYPOS(4,31)],ATT);
                        STDETT := EQU7(SDIGA[3]);
                        STR(STDETT:5:3,SORC);
                        LIMITCHECK(STDETT,37.0,37.5,38.5,39.0,SORC,ATT);
                        PUTOUT(SORC,FRAME[XYPOS(4,51)],ATT);
                    end;
                    SCMDE := SDIGA[7] ;
                    sorc := hexdig[scmde shr 12]
                            + hexdig[(scmde shr 8) and 15]
```

```
                                  + hexdig[(scmde shr 4) and 15]
                                  + hexdig[scmde and 15] ;
                          PUTOUT(SORC,FRAME[XYPOS(14,11)],GREEN);
                          SSTAT := SDIGA[6] ;
                          STR(SSTAT:5,SORC);
                          PUTOUT(SORC,FRAME[XYPOS(16,11)],GREEN);
                          SAPOSL := SDIGA[5];
                          SORC := SAREF(SAPOSL);
                          SAPOSH := SDIGA[4] ;
                          PUTOUT(SORC,FRAME[XYPOS(14,31)],GREEN);
                          SAPOS :=((SAPOSH AND 255) SHL 4 )
                                  OR ((SAPOSL AND 240) SHR 4);
                          STR(SAPOS:6,SORC);
                          PUTOUT(SORC,FRAME[XYPOS(15,31)],GREEN);
                          SAZP := SAPOS * 0.075 ;
                          STR(SAZP:6:3,SORC);
                          PUTOUT(SORC,FRAME[XYPOS(16,31)],GREEN);
                      END;
       else i := i ;   { do nothing }
            END;   { OF CASE }
       END;  { OF PROCEDURE PROCESSSCANNERDIGA }


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

       BEGIN
             CASE (NDIGA AND 15) OF

       1:   BEGIN
            NEPOS := '180 DEG    ' ;
            END;

       9:   BEGIN
            NEPOS := '180 WINDOW' ;
            END;

       3:   BEGIN
            NEPOS := '78 DEG     ' ;
            END;

       11:  BEGIN
            NEPOS := '78 WINDOW ' ;
            END;

       5:   BEGIN
            NEPOS := '0 DEG      ' ;
            END;

       13:  BEGIN
            NEPOS := '0 WINDOW   ' ;
```

```
        END;

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


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

    4:      BEGIN
            NAREF := '0 WINDOW   ';
            END;
    7:      BEGIN
            NAREF := '0 DEG      ';
            END;

    8:      BEGIN
            NAREF := '90 WINDOW ';
            END;

    11:     BEGIN
            NAREF := '90 DEG     ';
            END;

    12:     BEGIN
            NAREF := '180 WINDOW';
            END;

    15:     BEGIN
            NAREF := '180 DEG    ';
            END;

    ELSE
            NAREF := 'UNDEFINED ' ;

    END; {OF CASE }
    END; {OF NAPOS FUNCTION}


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

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

```
            BEGIN
                 CASE MINORFRAMENUM MOD 8 OF
                 3: BEGIN
                         STR(NDIGA:5,SORC);     {NMFTCH}
                         PUTOUT(SORC,FRAME[XYPOS(22,12)],GREEN);
                    END;
                 4: BEGIN
                         STR(NDIGA:5,SORC);     {NMSCH}
                         PUTOUT(SORC,FRAME[XYPOS(22,32)],GREEN);
                    END;
                 5: BEGIN
                         STR(NDIGA:5,SORC);       {NWTCH}
                         PUTOUT(SORC,FRAME[XYPOS(22,52)],GREEN);
                    END;
                 6: BEGIN
                         STR(NDIGA:5,SORC);       {NWSCH}
                         PUTOUT(SORC,FRAME[XYPOS(22,71)],GREEN);
                    END;
                 7: BEGIN
                         STR(NDIGA:5,SORC);        {NSMCH}
                         PUTOUT(SORC,FRAME[XYPOS(29,12)],GREEN);
                    END;
                 END; { OF INNER CASE }
            END;
  0:        BEGIN
                 SORC := HEXDIG[NDIGA SHR 12]
                       + HEXDIG[(NDIGA SHR 8) AND 15]
                       + HEXDIG[(NDIGA SHR 4) AND 15]
                       + HEXDIG[NDIGA AND 15] ;
           {     STR(NDIGA:5,SORC);    }     { COMMAND ECHO }
                 PUTOUT(SORC,FRAME[XYPOS(36,52)],GREEN);
            END;
  1:        BEGIN
                 STR(NDIGA:5,SORC);          { INSTRUMENT STATUS }
                 PUTOUT(SORC,FRAME[XYPOS(37,52)],GREEN);
            END;
  2:        BEGIN
                 SORC := NEPOS(NDIGA);         { NEPOS }
                 PUTOUT(SORC,FRAME[XYPOS(32,30)],GREEN);
            END;
  10,90:    BEGIN
                 SORC := NAREF(NDIGA) ;        { NAREF }
                 PUTOUT(SORC,FRAME[XYPOS(34,30)],GREEN);
                 NPOSIT := NDIGA SHR 4 ;
                 STR(NPOSIT:5,SORC);
                 PUTOUT(SORC,FRAME[XYPOS(35,30)],GREEN);    { NAPOS }
                 POSIT:= NPOSIT * 0.075 ;
                 STR(POSIT:10:3,SORC);                      { NADEG }
                 PUTOUT(SORC,FRAME[XYPOS(36,30)],GREEN);
            END;
```

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

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

                PUTOUT(SORC,FRAME[XYPOS(27,71)],GREEN);
          END;
25,105:   BEGIN
                NS := EQU11FM(NDIGA);
                STR(NS:5:2,SORC);      { NSWCST   }
                LIMITCHECK(NS,0.0,10.0,30.0,40.0,SORC,ATT);
                PUTOUT(SORC,FRAME[XYPOS(26,71)],ATT);
          END;
26,106:   BEGIN
                NS := EQU12FM(NDIGA);
                STR(NS:5:2,SORC);      { NMFSHT }
                LIMITCHECK(NS,10.0,20.0,40.0,50.0,SORC,ATT);
                PUTOUT(SORC,FRAME[XYPOS(25,32)],ATT);
          END;

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

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

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

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

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

42,122:         BEGIN
                        NS := EQU12FM(NDIGA);
                        STR(NS:5:2,SORC);      { NWFSHT }
                        LIMITCHECK(NS,10.0,20.0,40.0,50.0,SORC,ATT);
                        PUTOUT(SORC,FRAME[XYPOS(25,71)],ATT);
                END;

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

58,138:         BEGIN
                        NS := EQU12FM(NDIGA);
                        STR(NS:5:2,SORC);      { NMFSPT }
                        LIMITCHECK(NS,0.0,10.0,30.0,40.0,SORC,ATT);
                        PUTOUT(SORC,FRAME[XYPOS(26,12)],ATT);
                END;

64:             BEGIN
                        NS := EQU11FM(NDIGA);
                        STR(NS:5:2,SORC);      { NMFSAT   }
                        LIMITCHECK(NS,10.0,20.0,40.0,50.0,SORC,ATT);
                        PUTOUT(SORC,FRAME[XYPOS(24,32)],ATT);
                END;

65:             BEGIN
                        NS := EQU11FM(NDIGA);
                        STR(NS:5:2,SORC);     {NMFSLT   }
                        LIMITCHECK(NS,-10.0,0.0,40.0,50.0,SORC,ATT);
                        PUTOUT(SORC,FRAME[XYPOS(23,32)],ATT);
                END;

66,146:         BEGIN
                        NS := EQU12FM(NDIGA);
                        STR(NS:5:2,SORC);      {  NMFBBT }
                        LIMITCHECK(NS,0.0,10.0,55.0,60.0,SORC,ATT);
                        PUTOUT(SORC,FRAME[XYPOS(27,12)],ATT);
                END;
73:             BEGIN
                        NS := EQU13(NDIGA);
                        STR(NS:6:4,SORC);          { NCHTRV }
                        PUTOUT(SORC,FRAME[XYPOS(27,32)],GREEN);
                END;

74,154:         BEGIN
```

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

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

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

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

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

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

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

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

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

      153:        BEGIN
                        NS := EQU11FM(NDIGA);
                        STR(NS:5:2,SORC);   { NBEBT }
                        LIMITCHECK(NS,-10.0,0.0,40.0,50.0,SORC,ATT);
                        PUTOUT(SORC,FRAME[XYPOS(26,32)],ATT);
                  END;
      else i := i ;   { do nothing }
      END;   { OF CASE }
      END;      { OF ProcessNonScannerDIGA }

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

      0:    begin
            sorc := '                              ';
            att := green;
            end;

      1:    begin
            sorc := 'protocol error ' ;
            att := red ;
            end;

      2:    begin
            sorc := 'checksum error ';
            att := red ;
            end;

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

      end;

      procedure SCRDUMP(var i,j: integer) ;
      { 80 COLUMN BY 50 ROW SCREEN DUMP TO PRINTER }
      TYPE CHARBUFF = ARRAY[0..8000] OF CHAR ;
      VAR CFRAME: CHARBUFF ABSOLUTE $B800:$0000;
      PRFRAME: ARRAY[0..4000] OF CHAR ;
      K ,l: INTEGER ;
      begin
      IF (I+J = 0) THEN
```

```
BEGIN
FOR K := 0 TO 3939 DO
BEGIN
PRFRAME[K] := CFRAME[K*2];
END;
END;
for l:= 0 to 4 do
begin
   if (j<79)then
      WRITE(LST,PRFRAME[I*80 +j]) else writeln(lst,PRFRAME[I*80+j]);
   j:= j+1;
end;
if (j>79) then
begin
j:=0;
i := i+1;
if (i>45) then
begin
i := 0;
statpr :=0;
end;
end;
end;

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

     for j := 0 to 4 do
     begin
     k := 0 ;
     for i := 0 to 5 do
     begin
     line := ' ' ;
     for l:= 0 to 19 do
     begin
        bit := minorframe[j,k];
        line := line + hexdig[bit shr 4] + hexdig[bit and $f] + '  ' ;
        k := k + 1 ;
     end;
         writeln(lst,line);
    end;
         writeln (lst );
    end;
        statpr := 1 ;
end;

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

```
BEGIN
    FOR I := 0 TO 4 DO
    BEGIN
         FOR J:= 0 TO 103 DO
         MINORFRAME[I,J] := 0 ;
   END;
END;

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

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

BEGIN
     SCID := 0 ; { INITIALIZE SPACE CRAFT ID'S }
     OLDSCID := 0 ;{ THESE TWO VARIABLES ARE USED TO DETECT A
     CHANGE IN SATELLITE ID, THUS ENABLING A CLEARING OF THE
     SCREEN FOR THE NEW SATELLITE BEING DISPLAYED }
     ESIPL := OFF ;    { INITIALIZE SCANNER PULSE LOAD TO OFF }
     ASSIGN (F9,NOAA9);
     RESET(F9);
     eoffile(seg(F9),ofs(F9));
     ASSIGN(F10,NOAA10);
     RESET(F10);
     EOFFILE(SEG(F10),OFS(F10));
     STORINT[0] := SCRNINT[0] ;        { SAVE PRINT SCREEN VECTOR }
     STORINT[1] := SCRNINT[1] ;
     SCRNINT[0] := OFS(PRSTAT) ;    { PLACE NEW VECTOR TO OUR ROUTINE }
     SCRNINT[1] := CSEG ;
     NEWSCREEN ;                     { SET UP 80 X 50 DISPLAY SCREEN }
     DISPLAYACRO ;                       { DISPLAY ACRONYMS ON SCREEN }
     STATPR := 0;  linenum := 0; charnum:= 0;
     repeat
          INITRCV ;
          STAT;
     key := GETKEY ;
     if (key = 24 ) then status := 0 ;
```

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

REEP:     GETBUF ;
          STAT;
          key := GETKEY ;
          if (key = 2 ) then goto stop ;   { CONTROL B }
          if (key = 18 ) then  displayacro ; { CONTROL R }
          if (key = 4 ) then dumpbuff ;      { CONTROL D }
          IF (STATUS <> 0 ) THEN GOTO REEP ;
          { IF WE HAVE A BAD STATUS, THEN TRY AGAIN }
          IF (LENG = 0 ) THEN GOTO REEP ;
          { IF WE HAVE NO DATA, TRY AGAIN }
          FOR MN := 0 TO 4 DO
          { WE HAVE GOOD DATA, SO PROCESS 5 MINOR FRAMES }
          BEGIN
               DIVY(MN) ; { EXTRACT USEFUL DATA FROM TIP MINOR FRAME MN }
               if (minorframenum >319) then goto bottom ;
               IF ((FORG<>' 9') AND (FORG <> '10')) THEN GOTO BOTTOM ;
               ProcessDigb ;
               ProcessAnalog ;
               ProcessScannerDiga;
               ProcessNonscannerDiga;
               str(minorframenum:5,sorc);  {DISPLAY MINOR FRAME NUMBER }
               putout(sorc,frame[xypos(42,15)],white);
 bottom:       if statpr = 1 then scrdump(linenum,charnum);
          END;
          goto reep ;
STOP:     SCRNINT[0] := STORINT[0] ;
               {RESTORE NORMAL PRINT SCREEN VECTOR }
          SCRNINT[1] := STORINT[1] ; { DITTO }

          CLOSE(F9); { CLOSE NOAA-9 FILE }
          CLOSE(F10); { CLOSE NOAA-10 FILE }
          OLDSCREEN ; { CHANGE SCREEN MODE BACK TO NORMAL }
       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.

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

written by:    William L. Edmonds
               STX Corp.


}
LABEL KEEP ,stop,bottom;
TYPE ABC = STRING[80] ;
TYPE BYTEBUFF = ARRAY[0..8000] OF BYTE ;
TYPE integBUFF = ARRAY[0..4000] OF INTEGER ;
TYPE BITE = ARRAY[0..80] OF BYTE ;
TYPE MinorFrameBuffs = Array[0..103] of byte ;
Type dualbuffs = Array[0..4] of MinorFrameBuffs  ;
TYPE MinorFramewords = Array[0..51] OF INTEGER ;
Type dualwordbuffs = Array[0..4] of MinorFRamewords ;
TYPE  ERBREC = ARRAY[0..21] OF BYTE ;  { USED IN SAVING ERBE DATA TO DISK
TYPE  REGPACK = RECORD
                  ax,bx,cx,dx,bp,di,si,ds,es,flags:integer;
                END;
TYPE COMBUFF = ARRAY[0..519] OF BYTE ;
type smallbufw = array[0..3] of integer ;
TYPE COMBUFFW = ARRAY[0..259] OF INTEGER ;
CONST
     HEXDIG : ARRAY[0..15] OF CHAR = '0123456789ABCDEF' ;
     ON : STRING[3] = ' ON' ;
     OFF : STRING[3] = 'OFF' ;

     WHITE: INTEGER = 15 ;
     RED: INTEGER = 12 ;
     YELLOW : INTEGER = 14 ;
     GREEN : INTEGER = 10 ;
     msgrcv : byte = 2 ;
     xmtrcv : byte = 7 ;
     maxsec : integer = 10 ;
     sync : byte = $32 ;
     bisync : integer = $220 ;
     noerr : integer = 0 ;
     ptlerr : integer = 1 ;
     cserr : integer = 2 ;
     timerr : integer = 3 ;
     id0msg : array[0..3] of byte = ($32,$32,0,$8) ;
     id1msg : array[0..3] of byte = ($32,$32,0,$c) ;
     idmsln : integer = 4 ;
VAR
    spacecount : integer ;
    Spacelook : integer;
```

```
      Sample9,sample10 : integer ;
      Sample71 : integer ;
      Sample74 : integer ;
      days,hrs,mins :integer  ;   millisecs,secs :real ;
    ESIPL : ABC ;
    ERBDAT : ERBREC ;    { THIS ARRAY HOLDS ERBE DATA TO BE SAVED TO DISK }
    FL : FILE OF ERBREC ;
    key : integer ;
    state : byte ;
    SCID : BYTE ;     { SCID SHOULD BE = HEX "D" }
    id : byte ;
    IDNUM : INTEGER ;
    msgid : byte ;
    elpmin : byte ;
    elpsec : byte ;

    RECPACK :  REGPACK ;
    ah,al,ch,cl,dh : byte ;
    TIME : ARRAY[0..5] OF BYTE ;
    MN : INTEGER ;
    status,count: integer ;
    linenum,charnum: integer ;
    MinorFrameNUM : integer ;
    MajorFrameNUM : integer ;
    Digb : byte;
    scrnmode: array[0..15] of byte ;
    SDIGA : ARRAY[0..7] OF INTEGER ;
    NDIGA: INTEGER ;
    ANALOG : byte ;
    analogint : integer ;
    RANALOG : REAL ;

    VALST : ABC ;
    LENG : INTEGER ;
    RCVBUF : COMBUFF ABSOLUTE $8F80: $0000 ;
    RCVBUFW : COMBUFFW ABSOLUTE $8F80:$0004 ;
    msgbuf : smallbufw absolute $8F80: $0000 ;
    XMTBUF : COMBUFF ABSOLUTE $8F80: $0400 ;
    MinorFrame : dualbuffs absolute $7f80 : $0004 ;    { 9800:0004 ;  }
    MinorFrameW : dualwordbuffs absolute $7f80 : $0004  ;  { 9800:0004 ; }
    MINFW : COMBUFFW ABSOLUTE $7f80 : $0004  ; { 9800:0004 ;}
  FRAME: integBUFF ABSOLUTE $B800:$0000;
    BFRAME: BYTEBUFF ABSOLUTE $B800:$0000;
    txt : text ;
    txtfile : string[10] ;
    SORC:ABC ;
    ATT, XY: INTEGER;   I,K,   J:INTEGER;
    SCRNINT: ARRAY[0..1] OF INTEGER ABSOLUTE $0000:$0014;
    STORINT : ARRAY[0..1] OF INTEGER ;
    statpr : byte absolute $0050:$0000;
```

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

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



PROCEDURE S5080(var  i :byte); EXTERNAL 'CONO.COM';
PROCEDURE PUTOUT(VAR SORC:ABC;VAR FRAME:INTEGER;ATTR:INTEGER);
 EXTERNAL 'PUTOUT.COM';
FUNCTION PRSTAT:INTEGER; EXTERNAL 'PRSTAT.COM';


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

PROCEDURE OUTPUT(VAR SORC: ABC ; VAR FRAME : INTEGER ; ATTR : INTEGER );
VAR BLANKS : ABC ;
BEGIN
{       BLANKS := '           ' ;     11 BLANKS }
 {      PUTOUT(BLANKS, FRAME , WHITE );           }
        PUTOUT(SORC,FRAME,ATTR);
END;

function cntlb : integer ;
begin
     with recpack do
          begin
               ah := 6 ;
               al := 0;
               ax := ah shl 8 + al ;
               dx := $ff ;
          end;
          intr($21,recpack);
```

```
                with recpack do
                begin
                     al := ax and $ff ;
                end;
                cntlb := al ;
      end;



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

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



      procedure NEWSCREEN ;
      BEGIN
           SCRNMODE[0] := $71;
           SCRNMODE[1] := $50;
           SCRNMODE[2] := $5A;
           SCRNMODE[3] := $0F;
           SCRNMODE[4] := $1B;
           SCRNMODE[5] := 6;
           SCRNMODE[6] := $19;
           SCRNMODE[7] := $1A;
           SCRNMODE[8] := 3;
           SCRNMODE[9] := 7;
           SCRNMODE[10] := $20 ;
           SCRNMODE[11] := $20 ;
           SCRNMODE[12] := 0;
           SCRNMODE[13] := 0;
           SCRNMODE[14] := 0;
           SCRNMODE[15] := 0;
           S5080(SCRNMODE[0]);
           PORT[$3D8] := $19 ;
           PORT[$3E0] := $18 ;
           PORT[$3D9] := 0 ;


      END;

      PROCEDURE OLDSCREEN ;
      VAR LOC : INTEGER ;
```

```
BEGIN
      FOR LOC := 0 TO 3999 DO
      FRAME[LOC] := $F00 ;

      SCRNMODE[4] := $1F ;
      SCRNMODE[7] := $1C ;
      SCRNMODE[8] := 2;
      SCRNMODE[10] := 6;
      SCRNMODE[11] := 7;
      S5080(SCRNMODE[0]);
END;


PROCEDURE DisplayTime;
VAR TIMSTRING : ABC ;
BEGIN
    days := (time[0]shl 1) +((time[1] and 128)shr 7) ;
    millisecs := ((( time[1]and 7)*256.0 + time[2])*256.0 + time[3])*256.0
    hrs := trunc(millisecs/3600000.0) ;
    mins := trunc(millisecs/60000.0) mod 60 ;
    secs := trunc((millisecs/1000.0)-mins*60.0-hrs*3600.0);
    str(days:4,sorc); putout(sorc,frame[xypos(42,40)],white);
    str(hrs:2,sorc); putout(sorc,frame[xypos(42,45)],white);
    str(mins:2,sorc);  putout(sorc,frame[xypos(42,50)],white);
    str(secs:6:3,sorc); putout(sorc,frame[xypos(42,55)], white);
END;


PROCEDURE DIVY ;      { reads TIP MINOR FRAME data from disk}
LABEL RETURN ;
var nextmf,tipstatus: integer ;   nmf,mf: string[4] ;
BEGIN
      READ(FL,ERBDAT);
      nextmf := minorframenum +1 ;
      if (nextmf > 319 ) then nextmf := 0 ;
      MINORFRAMENUM := ERBDAT[1] + (ERBDAT[0] AND 1)shl 8;
      if (minorframenum<>nextmf) then
      begin
           str(nextmf:4,nmf);
           str(minorframenum:4,mf);
           sorc := 'expecting mf ' + nmf + ' but found mf ' + mf ;
           putout(sorc,frame[xypos(45,2)],yellow);
      end ;
      if (minorframenum = 0 ) then
      begin
           sorc := '                                        ';
           putout(sorc,frame[xypos(45,2)],yellow);
          FOR I := 0 TO 4 DO
```

```
                TIME[I] := ERBDAT[I+2] ;
                displaytime;
            END;
            DIGB := ERBDAT[6];
            ANALOG := ERBDAT[7] ;
            analogint := analog ;
            NDIGA := (ERBDAT[14] shl 8) or ERBDAT[15] ;
            SDIGA[0] := (ERBDAT[8] shl 8 or ERBDAT[9] )SHR 4 ;
            SDIGA[1] := (((ERBDAT[9] AND 15) SHL 8) OR ERBDAT[10]);
            SDIGA[2] := (ERBDAT[11] SHL 4) OR ( ERBDAT[12] SHR 4) ;
            SDIGA[3] := ((ERBDAT[12] and 15)shl 8 or ERBDAT[13] ) ;
            SDIGA[4] := (ERBDAT[16] shl 8 or ERBDAT[17] )SHR 4;
            SDIGA[5] := (((ERBDAT[17] AND 15 ) SHL 8) OR ERBDAT[18]);
            SDIGA[6] := (ERBDAT[19] SHL 4) OR (ERBDAT[20] SHR 4 ) ;
            SDIGA[7] :=(( ERBDAT[20] and 15) shl 8 or ERBDAT[21] ) ;
            ssch := sdiga[0] ;
            slch := sdiga[1] ;
            stch := sdiga[2] ;
            scpos := sdiga[3] ;
      RETURN:
      END;  { OF DIVY }


      Procedure ScanPos ;

      Begin
           Case Minorframenum mod 40 of

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

      }
               Spacelook := Spacelook + Sdiga[3] + Sdiga[7] ;
               spacecount := spacecount + 2 ;
              end;
      3 :     begin
      {
               Complete the sum of space look data and calculate average.

      }
               Spacelook := Spacelook + Sdiga[3] + Sdiga[7];
               spacecount := spacecount + 2 ;
               Spacelook := Spacelook div 8 ;
               sorc := 'SPACE '; output(sorc,frame[xypos(17,2)],white);
               str(spacelook:5,sorc) ; output(sorc,frame[xypos(17,11)],green);
              end;

      4 :     begin
      {
              Get first and second Earth scan positions.
```

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

35:       begin
{
           Get first internal cal position.

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

36:       begin
{
           Get 4th internal cal position.

}
           Sample74 := Sdiga[7] ;
           sorc := 'Sample 74 ' ; output(sorc,frame[xypos(17,61)],white);
           str(sample74:5,sorc) ; output(sorc,frame[xypos(17,71)],green);
           writeln(lst,days:5,hrs:5,mins:3,secs:6:2,spacecount:4,
           spacelook:10,sample9:10,sample10:10,sample71:10,sample74:10);
           spacelook := 0 ;
           spacecount := 0 ;
          end;
else   begin end;
end;  { of case }
end;  { of procedure  }




PROCEDURE DISPLAYACRO ;
VAR I: INTEGER;
BEGIN
 txtfile := 'sorc.txt' ;
 assign(txt,txtfile);
 reset(txt);
 att := 15 ;
i := 80 ;
while not eof(txt) do
begin
readln(txt,sorc);
sorc := sorc + '                ';
putout(sorc,frame[i],att);
```

```
i := i + 80 ;
end;
close (txt);

END;
procedure forwardjump ;
var ipos : integer ;
begin
      ipos := filepos(fl) ;
      ipos := ipos + 100 ;
      seek(fl,ipos) ;

end ;


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



BEGIN
     ESIPL := 'OFF' ;
WRITELN(' ENTER SPACE CRAFT ID (9 OR 10) ');
READLN(IDNUM);
IF IDNUM = 9 THEN ASSIGN(FL,'NOAA9.DAT')
ELSE ASSIGN(FL,'NOAA10.DAT');
     RESET(FL);
     STORINT[0] := SCRNINT[0] ;          { SAVE PRINT SCREEN VECTOR }
     STORINT[1] := SCRNINT[1] ;
     SCRNINT[0] := OFS(PRSTAT) ;       { PLACE NEW VECTOR TO OUR ROUTINE }
     SCRNINT[1] := CSEG ;
     NEWSCREEN ;                       { SET UP 80 X 50 DISPLAY SCREEN }
     DISPLAYACRO ;                      { DISPLAY ACRONYMS ON SCREEN }
     STATPR := 0;  linenum := 0; charnum:= 0;
     spacecount := 0 ;
     spacelook := 0 ;
     MINORFRAMENUM := -1 ;
WRITELN(LST,' SCAN CHECK FOR NOAA-',IDNUM:2);
repeat

REEP:

        key := cntlb ;
        if (key = 2 ) then goto stop ;
        if (key = 18 ) then  displayacro ;
{        if (key = 6 ) then forwardjump ;  }
        if (key = 11 ) then
        begin
             repeat
                   key := cntlb ;
                   if key = 7 then goto reep ;
             until key = 11 ;
```

```
                 rewrite(fl) ;
                 goto stop ;
            end;
            if (key = 16 ) then
            repeat
                  key := cntlb ;
            until key = 7 ;


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

    { Process scan position data }
                 Scanpos ;

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

    until eof(fl) ;
    STOP:   SCRNINT[0] := STORINT[0] ;
            SCRNINT[1] := STORINT[1] ;
            CLOSE(FL);
            OLDSCREEN ;
          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.ESMCAPR
.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.
FTN5,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
C
C
C
C     WRITTEN BY : SUDHA NATARAJAN
C     SYSTEM     : NOS CDC
C     PROJECT    : ERBE
C                  ST SYSTEMS CORPORATION
C                  HAMPTON VA-23666.
C
C     THIS IS ERBS SANNER MAM ALIBRATION PROGRAM.
C     THIS PROGRAM IS USED TO OBTAIN SOLAR CALIBRATION DATA FOR ALL C
C     THE THREL RADIOMETRIC CHANNEL DURING A SOLAR TRANSIT.         C
C     THE SHORTWAVE AND TOTAL CHANNELS USE THE MAM (MIRROR          C
C     ATTENUATOR MOSAIC) AS A PRIMARY CALIBRATION DURING A SOLAR    C
C     VIEW                                                          *
C    *
C
C     ROUTINES USED       : BUFFER IN,SPREAD
C     SUBROUTINES CALLED : TIMM,CTEU,CHPLT
C
C     CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C           DLT60L   SLCH SAMPLE 60 OUTPUT                          *
C           DLT60S   SSCH SAMPLE 60 OUTPUT                          *
C           DLT60T   STCH SAMPLE 60 OUTPUT                          *
C           DLT73L   SLCH SAMPLE 73 OUTPUT                          *
C           DLT73S   SSCH SAMPLE 73 OUTPUT                          *
C           DLT73T   STCH SAMPLE 73 OUTPUT                          *
C           IBUF     ERBS RECORD IN LARC FORMAT                     *
C           IBYTES   ERBS RECORD WHEN 'SPREAD' ROUTINE IS USED      *
C           IDAY     THE DAY NUMBER OF ORBITAL DATA BEING ANALYSED   *
C           IECHO    COMMAND ECHO                                   *
C           IYEAR    YEAR YHE DATA IS RECORDED                      *
C           K        INDEX OF THE TEMP. REF VOLTAGE IN EACH MF      *
C           L        INDEX OF THE MAM TEMP. IN EACH MF              *
C           M        INDEX IF THE TOTAL BLACK BODY TEMP.IN EACH MF  *
C           N        INDEX OF THE DETECTOR TEMP. IN EACH MF         *
C           SCH3     AVERAGE OF SSCH SAMPLE 3 OF 4 SCANS            *
C           SCH60    AVERAGE OF SSCH SAMPLE 60 OF 4 SCANS           *
C           SCH73    AVERAGE OF SSCH SAMPLE 73 OF 4 SCANS           *
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C               SLCH3     AVERAGE OF SLCH SAMPLE 3 OF 4SCANS                    *
C               SLCH60    AVERAGE OF SLCH SAMPLE 60 OF 4 SCANS                  *
C               SLCH73    AVERAGE OF SLCH SAMPLE 73 OF 4 SCANS                  *
C               SLDETT    LONGWAVE DETECTOR TEMP IN ENG. UNITS                  *
C               SSDETT    SHORTWAVE DETECTOR TEMP IN ENG.UNITS                  *
C               SSMAMT    SHORTWAVE MAM TEMP. IN ENG.UNITS                      *
C               SSMBT     SHORTWAVE MAM BAFFLE TEMP IN ENG.UNITS                *
C               STBBT     TOTAL BLACK BODY TEMP IN ENG.UNITS                    *
C               STDETT    TOTAL DETECTOR TEMP IN ENG UNITS                      *
C               STMAMT    TOTAL MAM TEMP IN ENG. UNITS                          *
C               STMBT     TOTLA MAM BAFFLE TEMP IN ENG UNITS                    *
C               STCH3     AVERAGE OF STCH SAMPLE 3 OF 4 SCANS                   *
C               STCH60    AVERAGE OF STCH SAMPLE 60 OF 4 SCANS                  *
C               STCH73    AVERAGE OF STCH SAMPLE 73 OF 4 SCANS                  *
C     CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
       COMMON/MAM/SSMAMT(130),SSMBT(130),STMAMT(130),STMBT(130),
      *STBBT(130),SSDETT(130),SLDETT(130),STDETT(130),STRV(130)
       COMMON/TIM/JDAY(130),JHR(130),JMIN(130),JSEC(130)
       COMMON/FTIM/FYEAR,FDAY,FHR,FMIN,FSEC,FHRE,FMINE,FSECE
       INTEGER ACCM3,ACCM60,ACCM73,BCCM3,BCCM60,BCCM73,DCCM3,DCCM60,
      *DCCM73
       DIMENSION IBUF(402),IBYTES(3015),SCH3(130),SCH60(130),
      *SCH73(130),SLCH3(130),SLCH60(130),SLCH73(130),STCH3(130),
      *STCH60(130),STCH73(130),X(130),DLT60S(130),DLT73S(130),
      *DLT60L(130),DLT73L(130),DLT60T(130),DLT73T(130),IECHO(130)
       IP=0
       IPOS3=58
       IPOS60=536
       IPOS73=682
       MAMT=700
       ITBB=716
       IDET=719
       ITRV=710
C
C      BEGIN BUFFERING IN THE RECORD UNTIL A VALID RECORD IS
C      ENCOUNTERED.
C      USE SPREAD ROUTINE TO STORE 8 BITS IN CDC 60 BIT WORD.
C      CALL TO SUBROUTINE 'TIMM' TO GET THE TIME
C
1      BUFFER IN(1,1)(IBUF(1),IBUF(402))
       IF(UNIT(1))2,500,500
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
    2     BUFFER IN(1,1)(IBUF(1),IBUF(402))
          IF(UNIT(1))22,500,500
   22     K=LENGTH(1)
          CALL SPREAD(IBUF(1),IBYTES(1),8,60,402,N)
          CALL TIMM(IBYTES,ISEC,IYEAR,KHR,KMIN,KSEC,IDAY)
C
C         INITIALISE THE ACCUMULATORS TO ZERO
C
          ACCM3=0
          ACCM60=0
          ACCM73=0

          BCCM3=0
          BCCM60=0
          BCCM73=0
C
          DCCM3=0
          DCCM60=0
          DCCM73=0
          NSMSM=0
          NSMSB=0
          NSMTM=0
          NSMTB=0
          NSMTBB=0
          NSMSDT=0
          NSMLDT=0
          NSMTDT=0
          NSMTRV=0
          IP=IP+1
C
C         PROCESS EACH SCAN
C
          DO 200 I=1,4
          IPT3= (I-1)*752 + IPOS3
          IPT60 = (I-1)*752 + IPOS60
          IPT73 = (I-1)*752 + IPOS73
C
          L= (I-1)*752 + MAMT
          M= (I-1)*752 + ITBB
          N= (I-1)*752 + IDET
          K=(I-1)*752 + ITRV
C
```

```
C      GET THE DATA OF SAMPLE 3,60,73 OF THE 3 DETECTORS
C
       IA=SHIFT(IBYTES(IPT3+1),-4).AND.15
       IB=SHIFT(IBYTES(IPT3),+4)
       ISCH3=IB.OR.IA
       IA=IBYTES(IPT3+1).AND.15
       ILCH3=SHIFT((IA),+8).OR.IBYTES(IPT3+2)
       IA=SHIFT(IBYTES(IPT3+4),-4).AND.15
       IB=SHIFT(IBYTES(IPT3+3),+4)
       ITCH3=IB.OR.IA

       IA=SHIFT(IBYTES(IPT60+1),-4).AND.15
       IB=SHIFT(IBYTES(IPT60),+4)
       ISCH60=IB.OR.IA
       IA=IBYTES(IPT60+1).AND.15
       ILCH60=SHIFT((IA),+8).OR.IBYTES(IPT60+2)
       IA=SHIFT(IBYTES(IPT60+4),-4).AND.15
       IB=SHIFT(IBYTES(IPT60+3),+4)
       ITCH60=IB.OR.IA
C
       IA=SHIFT(IBYTES(IPT73+1),-4).AND.15
       IB=SHIFT(IBYTES(IPT73),+4)
       ISCH73=IB.OR.IA
       IA=IBYTES(IPT73+1).AND.15
       ILCH73=SHIFT((IA),+8).OR.IBYTES(IPT73+2)
       IA=SHIFT(IBYTES(IPT73+4),-4).AND.15
       IB=SHIFT(IBYTES(IPT73+3),+4)
       ITCH73=IB.OR.IA
C
C      GET THE TEMPERATURE DATA
C      GET,SHORTWAVE MAM,SHORTWAVE MAM BAFFLE TEMP,TOTAL MAM TEMP,
C      TOTAL BAFFLE TEMP.,TOTAL BLACK BODY TEMP.,SHORTWAVE DETECTOR
C      TEMP.,LONGWAVE DETECTOR TEMP.,TOTAL DETECTOR TEMP.
C
C
       IA= SHIFT(IBYTES(L),+4)
       IB= SHIFT(IBYTES(L+1),-4) .AND. 15
       ISMAM = IA .OR. IB
       IA = IBYTES(L+1) .AND. 15
       ISMBT = SHIFT((IA),+8) .OR. IBYTES(L+2)
       IA = SHIFT(IBYTES(L+3),+4)
       IB= SHIFT(IBYTES(L+4),-4) .AND. 15
```

```
      ITMAM = IA .OR. IB
      IA = IBYTES(L+4)  .AND. 15
      ITMBT = SHIFT((IA),+8) .OR. IBYTES(L+5)
      IA = IBYTES(M) .AND. 15
      ITBBT = SHIFT((IA),+8) .OR. IBYTES(M+1)
      IA = IBYTES(N) .AND. 15
      ISDET = SHIFT((IA),+8) .OR. IBYTES(N+1)
      IA = SHIFT(IBYTES(N+2),+4)
      IB= SHIFT(IBYTES(N+3),-4) .AND. 15
      ILDET = IA .OR. IB
      IA = IBYTES(N+3) .AND. 15
      ITDET = SHIFT((IA),+8) .OR. IBYTES(N+4)
      IA=IBYTES(K) .AND. 15
      ISTRV=SHIFT((IA),+8).OR.IBYTES(K+1)
C
      ACCM3 = ACCM3 + ISCH3
      ACCM60 = ACCM60 + ISCH60
      ACCM73 = ACCM73 + ISCH73
C
      BCCM3 = BCCM3 + ILCH3
      BCCM60 = BCCM60 + ILCH60
      BCCM73 = BCCM73 + ILCH73
C
      DCCM3 = DCCM3 + ITCH3
      DCCM60 = DCCM60 + ITCH60
      DCCM73 = DCCM73 + ITCH73
      NSMSM = NSMSM + ISMAM
      NSMSB = NSMSB + ISMBT
      NSMTM = NSMTM + ITMAM
      NSMTB = NSMTB + ITMBT
      NSMTBB = NSMTBB + ITBBT
      NSMSDT = NSMSDT + ISDET
      NSMLDT = NSMLDT + ILDET
      NSMTDT = NSMTDT + ITDET
      NSMTRV=NSMTRV + ISTRV
C
200   CONTINUE
      JDAY(IP)=IDAY
      JHR(IP)=KHR
      JMIN(IP)=KMIN
      JSEC(IP)=KSEC
C
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C     GET THE AVERAGE SAMPLE 3,60,73 OF THE DETECTORS OF 4 SCANS
C
      SCH3(IP) = FLOAT(ACCM3) / 4.
      SCH60(IP) = FLOAT(ACCM60) / 4.
      DLT60S(IP)=SCH60(IP)-SCH3(IP)
      SCH73(IP) = FLOAT(ACCM73) / 4.
      DLT73S(IP)=SCH73(IP)-SCH3(IP)
C
      SLCH3(IP) = FLOAT(BCCM3) / 4.
      SLCH60(IP) = FLOAT(BCCM60) / 4.
      DLT60L(IP)=SLCH60(IP)-SLCH3(IP)
      SLCH73(IP) = FLOAT(BCCM73) / 4.
      DLT73L(IP)=SLCH73(IP)-SLCH3(IP)
C
      STCH3(IP) = FLOAT(DCCM3) / 4.
      STCH60(IP) = FLOAT(DCCM60) / 4.
      DLT60T(IP)=STCH60(IP)-STCH3(IP)
      STCH73(IP) = FLOAT(DCCM73) / 4.
      DLT73T(IP)=STCH73(IP)-STCH3(IP)
      X(IP)=FLOAT(IP)
C
C     GET THE AVERAGE TEMP. DATA IN COUNTS OF 4 SCANS
C
      SSMAMT(IP) = FLOAT(NSMSM) / 4.
      SSMBT(IP) = FLOAT(NSMSB) / 4.
      STMAMT(IP) = FLOAT(NSMTM) / 4.
      STMBT(IP) = FLOAT(NSMTB) / 4.
      STBBT(IP) = FLOAT(NSMTBB) / 4.
      SSDETT(IP) = FLOAT(NSMSDT) / 4.
      SLDETT(IP) = FLOAT(NSMLDT) / 4.
      STDETT(IP) = FLOAT(NSMTDT) / 4.
      STRV(IP)=FLOAT(NSMTRV) / 4.
C
      IA=IBYTES(728).AND.15
      IECHO(IP)=SHIFT((IA),+8).OR.IBYTES(729)
C
C     READ ANOTHER RECORD
C
      GO TO 2
C
C
C     CALL TO SUBROUTINE CTEU TO CONVERT TO ENGINEERING UNITS
```

```
C
500   CALL CTFU(IP)
C
C     WRITE HEADR IN THE OUTPUT FILE
C
      WRITE(11,501)IYEAR,IDAY
501   FORMAT('1',22X,'ERBS  SCANNER  RADIOMETRIC  TABLE'/33X,
     *'TEST DAY  :',2X,I2,'/',I3,/'0',2X,'MAJOR FRAME',29X,
     *'POSITIONS'/6X,'TIME',10X,'DET',10X,'SPACE',10X,'MAM',
     *9X,'INT/CAL',10X,'DLTMAM',10X,'DLTINT/CAL'/2X,'HRS/MIN/SECS',
     *6X,'TYPE')
      ILINE=7
C
C     WRITE RADIOMETRIC DATA IN THE OUTPUT FILE
C
      DO 510 IPT=1,IP
      WRITE(11,511)JHR(IPT),JMIN(IPT),JSEC(IPT),SCH3(IPT),SCH60(IPT),
     *SCH73(IPT),DLT60S(IPT),DLT73S(IPT),SLCH3(IPT),SLCH60(IPT),
     *SLCH73(IPT),DLT60L(IPT),DLT73L(IPT),STCH3(IPT),
     *STCH60(IPT),STCH73(IPT),DLT60T(IPT),DLT73T(IPT),TECHO(IPT)
511   FORMAT(3X,I2,'/',I2,'/',I2,9X,'SSCH',9X,F7.2,6X,F7.2,7X,
     *F7.2,7X,F7.2,7X,F7.2/20X,'SLCH',9X,F7.2,6X,
     *F7.2,7X,F7.2,7X,F7.2,7X,F7.2/20X,'STCH',9X,F7.2,6X,
     *F7.2,7X,F7.2,7X,F7.2,7X,F7.2,7X,I6)
      ILINE=ILINE+3
      IF(ILINE.GT.32)THEN
      WRITE(11,501)IYEAR,IDAY
      ILINE=7
      ENDIF
510   CONTINUE

      WRITE(11,650)IYEAR,IDAY
650   FORMAT('1',22X,'ERBS SCANNER  TEMPERATURE TABLE'/33X,
     *'TEST DAY  :',2X,I2,'/',I3/'0',2X,'MAJOR FRAME'/6X,'TIME'/
     *2X,'HRS/MIN/SECS',2X,'SSMAMT',2X,'SSMBT',3X,'STMAMT',2X,'STMBT',
     *3X,'STBBT',3X,'SSDETT',2X,'SLDETT',2X,'STDETT')
      ILINE=7
C
C     WRITE TEMPERATURE DATA TO THE OUTPUT FILE
C
      DO 900 K=1,IP
      WRITE(11,800)JHR(K),JMIN(K),JSEC(K),SSMAMT(K),
```

1
1
1

```
              *SSMBT(K),STMAMT(K),STMBT(K),STBBT(K),
              *SSDETT(K),SLDETT(K),STDETT(K)
               ILINE=ILINE+1
               IF(ILINE.GT.32)THEN
 1             WRITE(11,650)IYEAR,IDAY
 1             ILINE=7
 1             ENDIF
         900   CONTINUE
         800   FORMAT(3X,I2,'/',I2,'/',I2,5X,8(F6.2,2X))
         C
         C     CALL TO SUBROUTINE CHPLT TO PLOT THE SPACE,MAM AND INT.CAL
         C     LOOK
         C
               CALL CHPLT(SCH3,DLT60S,DLT73S,SLCH3,DLT60L,DLT73L,
              *STCH3,DLT60T,DLT73T,IP,IYEAR,X)
               GO TO 4
         3     BUFFER IN(1,1)(IBUF(1),IBUF(402))
               IF(UNIT(1))2,4,4
         4     STOP
               END
```

ORIGINAL PAGE IS OF POOR QUALITY

--VARIABLE MAP--(LO=A)

-NAME-	--ADDRESS	--BLOCK	-----PROPERTIES	-------TYPE	---------SIZE	-NAME-	--ADDRESS	--BLOCK	-----PROPERTIES	-------TYPE	---------SIZE
ACCM3	706B			INTEGER		FHRE	5B	/FTIM/		REAL	
ACCM60	707B			INTEGER		FMIN	3B	/FTIM/		REAL	
ACCM73	710B			INTEGER		FMINE	6B	/FTIM/		REAL	
BCCM3	711B			INTEGER		FSEC	4B	/FTIM/		REAL	
BCCM60	712B			INTEGER		FSECE	7B	/FTIM/		REAL	
BCCM73	713B			INTEGER		FYEAR	0B	/FTIM/		REAL	
DCCM3	714B			INTEGER		I	13743B			INTEGER	
DCCM60	715B			INTEGER		IA	13751B			INTEGER	
DCCM73	716B			INTEGER		IB	13752B			INTEGER	
DLT60L	12500B			REAL	130	IBUF	717B			INTEGER	402
DLT60S	12074B			REAL	130	IBYTES	1541B			INTEGER	3015
DLT60T	13104B			REAL	130	IDAY	13731B			INTEGER	
DLT73L	12702B			REAL	130	IDET	13720B			INTEGER	
DLT73S	12276B			REAL	130	IECHO	13510B			INTEGER	130
DLT73T	13306B			REAL	130	ILCH3	13754B			INTEGER	
FDAY	1B	/FTIM/		REAL		ILCH60	13757B			INTEGER	
FHR	2B	/FTIM/		REAL		ILCH73	13762B			INTEGER	

DO=-LONG/-OT,ARG= COMMON/-FIXED,CS= USEP/-FIXED,DB=-TB/-SB/-SL/-ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=ESMCAL,L=LF.

```
      SUBROUTINE TIMM(IBYTES,ISEC,IYEAR,KHR,KMIN,KSEC,IDAY)
C
C     THIS SUBROUTINE GETS THE TIME OF THE ORBITAL DATA
C
C     CALLING ROUTINE  : ESMCAL
C
      DIMENSION IBYTES(3015)
      IA=IBYTES(2).AND.1
      IB=SHIFT((IA),+8).OR.IBYTFS(172)
      IC=SHIFT(IBYTES(173),-3).AND.31
C    ****GET THE JULIAN DAY
C
      JDAY=SHIFT((IB),+5).OR.IC
C
C    * 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
      ID=IBYTES(173).AND.7
      IE=SHIFT((ID),+8).OR.IBYTES(174)
      IF=SHIFT(IBYTFS(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)
      PETURN
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      SUBROUTINE CTEU(IP)
C
C     THIS SUBROUTINE GETS THE TEMPERATURE IN ENG.UNITS
C     USING COUNT CONVERSION EQUATION.
C
C
C     CALLING ROUTINE   :  ESMCAL
C
      COMMON/MAM/SSMAMT(130),SSMBT(130),STMAMT(130),STMBT(130),
     *STBBT(130),SSDETT(130),SLDETT(130),STDETT(130),STRV(130)
      DATA C1,C2,C3,C4,C5,C6/20.0091,.0129966,8.25284E-6,
     *2.33846E-8,1.67705E-11,3.99253E-15/
      DATA B1,B2,B3,B4,B5,B6/12.1752498,1.4411415E-2,5.04308321E-7,
     *2.62463814E-11,1.04124505E-14,8.22227099E-19/
      DATA D1,D2/36,9.766E-4/
      DATA S1/2.1106E-3/
      DO 100 I=1,IP
C
C     GET THE SHORTWAVE MAM TEMP
C
      SSMAMT(I) = -C1 +(C2 * SSMAMT(I))-(C3*SSMAMT(I)**2)+
     *(C4 * SSMAMT(I)**3) - (C5 * SSMAMT(I)**4)
     *+(C6 * SSMAMT(I)**5)
C
C     GET SHORTWAVE MAM BAFFLE TEMP
C
      SSMBT(I) = -C1 +(C2 * SSMBT(I))-(C3*SSMBT(I)**2)+
     *(C4 * SSMBT(I)**3) - (C5 * SSMBT(T)**4)
     *+(C6 * SSMBT(I)**5)
C
C     GET TOTAL MAM TEMP
C
      STMAMT(I) = -C1 +(C2 * STMAMT(I))-(C3*STMAMT(I)**2)+
     *(C4 * STMAMT(I)**3) - (C5 * STMAMT(I)**4)
     *+(C6 * STMAMT(I)**5)
C
C     GET TOTAL MAM BAFFLE TEMP
C
      STMBT(I) = -C1 +(C2 * STMBT(I))-(C3*STMBT(I)**2)+
     *(C4 * STMBT(I)**3) - (C5 * STMBT(I)**4)
```

```
      *+(C6 * STMBT(I)**5)
C
C      GET SHORTWAVE,LONGWAVE,TOTAL DETECTOR TEMP.
C
       SSDETT(I) = D1 + D2 * SSDETT(I)
       SLDETT(I) = D1 + D2 * SLDETT(I)
       STDETT(I) = D1 + D2 * STDETT(I)
       STRV(I) = -S1 * STRV(I)
       R=-6.40 /STRV(I)
C
C      GET TOTAL BLACK BODY TEMP
C
       STBBT(I) = STBBT(I) * R
       STBBT(I) = -B1 + (B2 * STBBT(I)) + (B3 * STBBT(I)**2) -
      *(B4 * STBBT(I)**3) + (B5 * STBBT(I)**4) -
      *(B6 * STBBT(I)**5)
 100   CONTINUE
       RETURN
       END
```


--VARIABLE MAP--(LO=A)

-NAME-	--ADDRESS	--BLOCK	-----PROPERTIES	-------TYPE	---------SIZE
B1	106B			REAL	
B2	107B			REAL	
B3	110B			REAL	
B4	111B			REAL	
B5	112B			REAL	
B6	113B			REAL	
C1	100B			REAL	
C2	101B			REAL	
C3	102B			REAL	
C4	103B			REAL	
C5	104B			REAL	
C6	105B			REAL	
D1	114B			REAL	
D2	115B			REAL	

-NAME-	--ADDRESS	--BLOCK	-----PROPERTIES	-------TYPE	---------SIZE
I	117B			INTEGER	
IP	1		DUMMY-ARG	INTEGER	
R	120B			REAL	
SLDETT	1414B	/MAM/		REAL	130
SSDETT	1212B	/MAM/		REAL	130
SSMAMT	0B	/MAM/		REAL	130
SSMBT	202B	/MAM/		REAL	130
STBBT	1010B	/MAM/		REAL	130
STDETT	1616B	/MAM/		REAL	130
STMAMT	404B	/MAM/		REAL	130
STMBT	606B	/MAM/		REAL	130
STRV	2020B	/MAM/		REAL	130
S1	116B			REAL	

DO=-LONG/-OT,ARG= COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/-ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=ESMCAL,L=LF.

```
      SUBROUTINE CHPLT(SCH3,DLT60S,DLT73S,SLCH3,DLT60L,DLT73L,
     *STCH3,DLT60T,DLT73T,IP,IYEAR,X)
C
C     THIS SUBROUTINE PLOTS THE RADIOMETRIC SAMPLES OF SPACE
C     MAM AND INTERNAL CAL LOOK
C
C     CALLING ROUTINE   :  ESMCAL
C
C
      CHARACTER DETLABL*4
      COMMON/TIM/JDAY(130),JHR(130),JMIN(130),JSEC(130)
      COMMON/FTIM/FYEAR,FDAY,FHR,FMIN,FSEC,FHRE,FMINE,FSECE
      DIMENSION SCH3(IP+2),DLT60S(IP+2),DLT73S(IP+2),X(IP+2)
      DIMENSION SLCH3(IP+2),DLT60L(IP+2),DLT73L(IP+2),STCH3(IP+2),
     *DLT60T(IP+2),DLT73T(IP+2)
C
C     SET THE ORIGIN AND SCALE FACTOR FOR THE X AXIS
C
      X(IP+1)=0.
      X(IP+2)=15.
      WRITE(11,150)X(IP+1),X(IP+2)
  150 FORMAT(2X,2(F7.2))
      FYEAR=FLOAT(IYEAR)
      FDAY=JDAY(1)
      FHR=JHR(1)
      FMIN=JMIN(1)
      FSEC=JSEC(1)
      FHRE=JHR(IP)
      FMINE=JMIN(IP)
      FSECE=JSEC(IP)
C
C     INITIALISE THE PLOTTING ROUTINE
C
      CALL PSEUDO
      CALL LEROY
      CALL ASCALE(SCH3,8.5,IP,1,10.)
      CALL ASCALE(DLT60S,8.5,IP,1,10.)
      CALL ASCALE(DLT73S,8.5,IP,1,10.)
      CALL CALPLT(2.,1.,-3)
C
```

```
C       DRAW THE AXES
C
        CALL AXES(0.,0.,0.,15.,X(IP+1),X(IP+2),1.,4.,
       *'RECORDS',.14,-7)
        CALL AXES(0.,0.,90.,8.5,SCH3(IP+1),SCH3(IP+2),1.,10.,
       *'SCPOS-3',.14,7)
        CALL AXES(0.,8.5,0.,15.,X(IP+1),X(IP+2),1.,4.,' ',0.0,1)
        CALL AXES(15.,0.,90.,8.5,SCH3(IP+1),SCH3(IP+2),1.,
       *10.,' ',0.0,-1)
        CALL NEWPEN (1)
C
C       PLOT THE SPACE DATA
C
        CALL LINPLT(X,SCH3,IP,1,0,0,1,1)
C
C       OFFSET THE PLOT WITHIN THE FRAME AND DRAW ANOTHER Y AXES
C
        CALL NEWPEN (2)
        CALL CALPLT(-1.,0.,-3)
        CALL AXES(0.,0.,90.,8.5,DLT60S(IP+1),DLT60S(IP+2),1.,10.,
       *'SCPOS-60',.14,8)
        CALL CALPLT(1.,0.,-3)
C
        CALL NEWPEN(2)
C
C       PLOT SSCH MAM DATA
C
        CALL LINPLT(X,DLT60S,IP,1,0,0,1,2)
C
C       OFFSET THE PLOT WITHIN THE FRAME AND DRAW 3RD Y AXES
C
        CALL NEWPEN (3)
        CALL CALPLT(16.,0.,-3)
        CALL AXES(0.,0.,90.,8.5,DLT73S(IP+1),DLT73S(IP+2),1.,10.,
       *'SCPOS-73',.14,8)
        CALL CALPLT(-16.,0.,-3)
C
        CALL NEWPEN(3)
C
C       PLOT SSCH INT-CAL DATA
C
        CALL LINPLT(X,DLT73S,IP,1,0,0,1,3)
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      CALL NEWPEN(1)
      DETLABL='SSCH'
C
C     CALL TO SUBROUTINE ELABL TO LABEL THE PLOT
C
      CALL ELABL(DETLABL)
C
C     ESTABLISH A NEW REFERENCE POINT FOR THE NEXT FRAME
C
      CALL NFRAME
C     CALL CALPLT(2.,1.,-3)
C     CALL ASCALE(SLCH3,8.5,IP,1,10.)
C     CALL ASCALE(SLCH60,8.5,IP,1,10.)
C     CALL ASCALE(SLCH73,8.5,IP,1,10.)
C     CALL AXES(0.,0.,0.,15.,X(IP+1),X(IP+2),1.,4.,
C    *'RECORDS',.14,-7)
C     CALL AXES(0.,0.,90.,8.5,SLCH3(IP+1),SLCH3(IP+2),1.,10.,
C    *'SCPOS-3',.14,7)
C     CALL AXES(0.,8.5,0.,15.,X(IP+1),X(IP+2),1.,4.,' ',0.0,1)
C     CALL AXES(15.,0.,90.,8.5,SLCH3(IP+1),SLCH3(IP+2),1.,
C    *10.,' ',0.0,-1)
C     CALL NEWPEN (1)
C     CALL LINPLT(X,SLCH3,IP,1,0,0,1,1)
C     CALL NEWPEN (2)
C     CALL CALPLT(-1.,0.,-3)
C     CALL AXES(0.,0.,90.,8.5,SLCH60(IP+1),SLCH60(IP+2),1.,10.,
C    *'SCPOS-60',.14,8)
C     CALL CALPLT(1.,0.,-3)
C     CALL NEWPEN(2)
C     CALL LINPLT(X,SLCH60,IP,1,0,0,1,2)
C     CALL NEWPEN (3)
C     CALL CALPLT(16.,0.,-3)
C     CALL AXES(0.,0.,90.,8.5,SLCH73(IP+1),SLCH73(IP+2),1.,10.,
C    *'SCPOS-73',.14,8)
C     CALL CALPLT(-16.,0.,-3)
C     CALL NEWPEN(3)
C     CALL LINPLT(X,SLCH73,IP,1,0,0,1,3)
C     CALL NEWPEN(1)
C     DETLABL='SLCH'
C     CALL ELABL(DETLABL)
C     CALL NFRAME
C
```

```
C     ESTABLISH A NEW REFERENCE POINT TO PLOT THE TOTAL CHANNEL
C
      CALL CALPLT(2.,1.,-3)
C
C     SCALE THE Y AXES AND STORE THE ORIGIN AND SCALE FACTOR
C
      CALL ASCALE(STCH3,8.5,IP,1,10.)
      CALL ASCALE(DLT60T,8.5,IP,1,10.)
      CALL ASCALE(DLT73T,8.5,IP,1,10.)
C
C     DRAW THE AXES
C
      CALL AXES(0.,0.,0.,15.,X(IP+1),X(IP+2),1.,4.,
     *'RECORDS',.14,-7)
      CALL AXES(0.,0.,90.,8.5,STCH3(IP+1),STCH3(IP+2),1.,10.,
     *'SCPOS-3',.14,7)
      CALL AXES(0.,8.5,0.,15.,X(IP+1),X(IP+2),1.,4.,' ',0.0,1)
      CALL AXES(15.,0.,90.,8.5,STCH3(IP+1),STCH3(IP+2),1.,
     *10.,' ',0.0,-1)
      CALL NEWPEN (1)
C
C     PLOT THE SPACE TOTAL CHANNEL
C
      CALL LINPLT(X,STCH3,IP,1,0,0,1,1)
      CALL NEWPEN (2)
C
C     OFFSET THE PLOT WITHIN THE FRAME
C
      CALL CALPLT(-1.,0.,-3)
      CALL AXES(0.,0.,90.,8.5,DLT60T(IP+1),DLT60T(IP+2),1.,10.,
     *'SCPOS-60',.14,8)
      CALL CALPLT(1.,0.,-3)
      CALL NEWPEN(2)
C
C     PLOT THE TOTAL MAM DATA
C
      CALL LINPLT(X,DLT60T,IP,1,0,0,1,2)
      CALL NEWPEN (3)
C
C     OFFSET THE PLOT WITHIN THE FRAME
C
      CALL CALPLT(16.,0.,-3)
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      CALL AXES(0.,0.,90.,8.5,DLT73T(IP+1),DLT73T(IP+2),1.,10.,
     *'SCPDS-73',.14,8)
      CALL CALPLT(-16.,0.,-3)
      CALL NEWPEN(3)
C
C     DRAW THE TOTAL INT-CAL DATA
C
      CALL LINPLT(X,DLT73T,IP,1,0,0,1,3)
      CALL NEWPEN(1)
      DETLABL='STCH'
C
C     CALL TO SUBROUTINE ELABL TO LABEL THE PLOT
C
      CALL ELABL(DETLABL)
      CALL NFRAME
C
C
C     TERMINATE THE GRAPH
C
      CALL CALPLT(0.0,0.0,999)
      RETURN
      END
```

ORIGINAL PAGE IS OF POOR QUALITY

--VARIABLE MAP--(LO=A)

-NAME---	ADDRESS	--BLOCK-----	PROPERTIES-------	TYPE---------	SIZE
DETLABL	1026B			CHAR*4	
DLT60L	5	DUMMY-ARG		REAL	ADJ-ARY
DLT60S	2	DUMMY-ARG		REAL	ADJ-ARY
DLT60T	8	DUMMY-ARG		REAL	ADJ-ARY
DLT73L	6	DUMMY-ARG		REAL	ADJ-ARY
DLT73S	3	DUMMY-ARG		REAL	ADJ-ARY
DLT73T	9	DUMMY-ARG		REAL	ADJ-ARY
FDAY	1B	/FTIM/		REAL	
FHR	2B	/FTIM/		REAL	
FHRE	5B	/FTIM/		REAL	
FMIN	3B	/FTIM/		REAL	
FMINE	6B	/FTIM/		REAL	
FSEC	4B	/FTIM/		REAL	

-NAME---	ADDRESS	--BLOCK-----	PROPERTIES-------	TYPE---------	SIZE
FSECE	7B	/FTIM/		REAL	
FYEAR	0B	/FTIM/		REAL	
IP	10	DUMMY-ARG		INTEGER	
IYEAR	11	DUMMY-ARG		INTEGER	
JDAY	0B	/TIM/		INTEGER	130
JHR	202B	/TIM/		INTEGER	130
JMIN	404B	/TIM/		INTEGER	130
JSEC	606B	/TIM/		INTEGER	130
SCH3	1	DUMMY-ARG		REAL	ADJ-ARY
SLCH3	4	DUMMY-ARG		REAL	ADJ-ARY
STCH3	7	DUMMY-ARG		REAL	ADJ-ARY
X	12	DUMMY-ARG		REAL	ADJ-ARY

```
DO=-LONG/-OT,ARG= COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/-ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=ESMCAL,L=LF.

              SUBROUTINE ELABL(DETLABL)
      C
      C
      C       THIS SUBROUTINE LABELS THE PLOTS
      C
      C
              CHARACTER DETLABL*(*)
              COMMON/FTIM/FYEAR,FDAY,FHP,FMIN,FSEC,FHRE,FMINE,FSECE
              CALL CHARACT(1.0,8.75,.10,'ERBS',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*(*)	
FDAY	1B	/FTIM/		REAL	
FHR	2B	/FTIM/		REAL	
FHRE	5B	/FTIM/		REAL	
FMIN	3B	/FTIM/		REAL	
FMINE	6B	/FTIM/		REAL	
FSEC	4B	/FTIM/		REAL	
FSECE	7B	/FTIM/		REAL	
FYEAR	0B	/FTIM/		REAL	

--PROCEDURES--(LO=A)

-NAME-	-----TYPE-----	--ARGS-	-----CLASS-----
CHARACT		7	SUBROUTINE
NUMBER		6	SUBROUTINE

ORIGINAL PAGE IS
OF POOR QUALITY

```
 BLOCKDATA COM        74/860  OPT=1,ROUND= A/ S/ M/-D,-DS     FTN 5.1+642         87/05/21. 08.21.17         PAGE      1
DO=-LONG/-OT,ARG= COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/-ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=ESMCAL,L=LF.

                     BLOCK DATA COM
                     COMMON/MAM/SSMAMT(130),SSMBT(130),STMAMT(130),STMBT(130),
                    *STBBT(130),SSDETT(130),SLDETT(130),STDETT(130),STRV(130)
                     COMMON/TIM/JDAY(130),JHR(130),JMIN(130),JSEC(130)
                     COMMON/FTIM/FYEAR,FDAY,FHR,FMIN,FSEC,FHRE,FMINE,FSECE
                     END

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

  FDAY         1B  /FTIM/                     REAL                   JSEC       606B  /TIM/                       INTEGER       130
  FHR          2B  /FTIM/                     REAL                   SLDETT    1414B  /MAM/                       REAL          130
  FHRE         5B  /FTIM/                     REAL                   SSDETT    1212B  /MAM/                       REAL          130
  FMIN         3B  /FTIM/                     REAL                   SSMAMT      0B  /MAM/                       REAL          130
  FMINE        6B  /FTIM/                     REAL                   SSMBT      202B  /MAM/                       REAL          130
  FSEC         4B  /FTIM/                     REAL                   STBBT     1010B  /MAM/                       REAL          130
  FSECE        7B  /FTIM/                     REAL                   STDETT    1616B  /MAM/                       REAL          130
  FYEAR        0B  /FTIM/                     REAL                   STMAMT     404B  /MAM/                       REAL          130
  JDAY         0B  /TIM/                      INTEGER        130     STMBT      606B  /MAM/                       REAL          130
  JHR        202B  /TIM/                      INTEGER        130     STRV      2020B  /MAM/                       REAL          130
  JMIN       404B  /TIM/                      INTEGER        130

--STATISTICS--

  PROGRAM-UNIT LENGTH               0B =   0
  CM LABELLED COMMON LENGTH      3242B =   1698
  CM STORAGE USED               61300B =   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.
FTN5,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.
```

```
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.

                  PROGRAM SOLCA
            C
            C
            C
            C
            C     AUTHOR : SUDHA NATARAJAN
            C     SYSTEM : NOS CDC
            C     PROJECT: ERBE
            C
            C
            C
            C     THIS SOFTWARE IS USED TO GET THE 'SOLAR CONSTANT' VALUES
            C     DETERMINED SIMULTANEOUSLY BY THE EARTH VIEWING AND
            C     SOLAR MONITOR ASSEMBLY(SMA) CHANNELS. SMA VALUES ARE USED
            C     AS STANDARDS. THE 'SOLAR CONSTANT' VALUES FROM EV CHANNELS
            C     ARE COMPARED TO SMA ONCE EVERY 2 WEEKS IN ORDER TO
            C     CALIBRATE/VALIDATE EARTH VIEWING CHANNELS.
            C
            C
            C
            C     CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
            C     A          ARRAY OF THE RADIOMETRIC CHANNEL DATA                C
            C     ADATE      CALIBRATION DATE                                     C
            C     ATIME      TIME OF SPACE LOOK OF THE EARTH CHANNELS             C
            C     AMTSP      MED-TOT AVE LIMIT TEMP(SPACE LOOK)                   C
            C     AMSSP      MED-SW AVE. LIMIT TEMP(SPACE LOOK)                   C
            C     AWTSP      WIDE-TOT AVE. LIMIT TEMP(SPACE LOOK)                 C
            C     AWSSP      WIDE-SW AVE. LIMIT TEMP(SPACE LOOK)                  C
            C     AMSPT
            C     B          ARRAY OF RADIOMETRIC CHANNEL DATA OF SOLAR VIEW      C
            C     BSD        STAN. DEV OF THE RAD.CHANNELS DURING SOLAR VIEW      C
            C     BTIME      SOLAR VIEW TIME OF THE EARTH VIEWING CHANNELS        C
            C     IDAY       THE DAY NUMBER                                       C
            C     BSTME      SUN LOOK OF THE SOL-MON DURING SHUTTER OPEN          C
            C     BMTSP      AVE LIMT TEMP OF MED-TOT DURING SUN LOOK             C
            C     BMSSP      AVE LIMT TEMP OF MED-SW DURING SUN LOOK              C
            C     BWTSP      AVE LIMT TEMP OF WID-TOT DURING SUN LOOK             C
            C     BWSSP      AVE LIMT TEMP OF WID-SW DURING SUN LOOK              C
            C     BMSPT      SLRPT TEMP OF MED-SW DURING SUN LOOK                 C
            C     BWSPT      SLRPT TEMP OF WID-SW DURING SUN LOOK                 C
```

ORIGINAL PAGE IS OF POOR QUALITY

```
C      BNMHT     HEAT SINK TEMP OF SOLAR MON DURING SUN LOOK         C
C      BNMAT     APER-TEMP OF SOLAR MON DURING SUN LOOK              C
C      BNMBT     BAFFLE TEMP OF SOLAR MON DURING SUN LOOK            C
C      SNANG(1)  SUN ANGLE OF THE EARTH CHANNELS  DURING SUN LOOK    C
C      SNANG(2)  SUN ANGLE OF SOLAR MON DURING SUN LOOK              C
C
C      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C
       COMMON/COEFF/RESTN(6),AREA(5),SWTRN(2)
C     ENTER THE FILGHT MODEL
C
C
       PRINT 100
100    FORMAT(10X,'1-NOAA-9'/10X,'2-ERBS'/10X,'3-NOAA-10'/10X,
      *'ENTER THE FLIGHT MODEL')
       READ *,IDF
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)
C
C
C      ENTER THE ACTIVE HEATER RESISTANCE,AREA OF THE PRIMARY
C      APERTURE,SHORTWAVE CHANNEL DOME FILTER TRANSMISSION.
C
C
C     ENTER THE RESISTANCE,AREA AND TRANSMISSION OF MED-TOT,
C     MED-SW,WID-TOT,WID-SW,SOLAR CHANNELS
C
C
       PRINT *,' ENTER THE MEDIUM TOTAL CHANNEL RESISTANCE  '
       READ *,RESTN(1)
       PRINT *,' ENTER THE AREA FOR THE MEDIUM TOTAL CHANNEL  '
       READ *, AREA(1)
       PRINT *,' ENTER THE MEDIUM SHORTWAVE CHANNEL RESISTANCE  '
       READ *,RESTN(2)
       PRINT *,' ENTER THE AREA FOR THE MEDIUM SHIRTWAVE CHANNEL '
       READ *,AREA(2)
       PRINT *,' ENTER THE WIDE TOTAL CHANNEL RESISTANCE  '
       READ *,RESTN(3)
       PRINT *,' ENTER THE AREA FOR THE WIDE TOTAL CHANNEL '
       READ *,AREA(3)
```

```
                  PRINT *,' ENTER THE WIDE SHORTWAVE CHANNEL RESISTANCE '
                  READ *,RESTN(4)
                  PRINT *,' ENTER THE AREA FOR THE WIDE SHORTWAVE CHANNEL '
                  READ *,AREA(4)
                  PRINT *,' ENTER THE SOLAR MONITOR CHANNEL RESISTANCE '
                  READ *, RESTN(5)
                  RESTN(6) = RESTN(5)
                  PRINT *,' ENTER THE AREA FOR THE SOLAR MONITOR CHANNEL'
                  READ *,AREA(5)
                  PRINT *,' ENTER THE TRANSMISSION FOR THE  MEDIUM SHORTWAVE CH'
                  READ *,SWTRN(1)
                  PRINT *,' ENTER THE TRANSMISSION FOR THE WIDE SHORTWAVE CH'
                  READ *,SWTRN(2)
            C
            C
            C     CALL TO SUBROUTINE PROC TO PROCESS THE RAW DATA TO IRRADIANCE
                  CALL PROC(IDF)
            C

                  STOP
                  END
```

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

  AREA        6B  /COEFF/                     REAL             5
  IDF       361B                              INTEGER
  RESTN       0B  /COEFF/                     REAL             6
  SWTRN      13B  /COEFF/                     REAL             2

--PROCEDURES--(LO=A)
 -NAME------TYPE---------ARGS------CLASS-----

  PROC                     1       SUBROUTINE
```

```
      SUBROUTINE PROC(IDF)
C
C
C     THIS SUBROUTINE PROC APPLIES RESISTANCE,AREA,TRANSMISSION
C     TO THE RAW DATA AND GETS THE SOLAR FLUX MEASUREMENTS
C     NORMALIZED TO 1 A.U.
C
      COMMON/COEFF/RESTN(6),AREA(5),SWTRN(2)
      COMMON/TVAR/AMTSP,BMTSP,AMSPT,BMSPT,AWTSP,BWTSP,AWSPT,BWSPT
      DIMENSION A(6),B(6),C(6),D(6),E(6),F(5),ASD(6),
     *BSD(6),SNANG(2),PRODCT(5)
      CHARACTER SPMOD*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,AWSPT,ANMHT,
     *ANMAT,ANMBT,IDAY,BTIME,(B(I),I=1,6),(BSD(I),I=1,6),
     *BSTME,BMTSP,BMSSP,BWTSP,BWSSP,BMSPT,BWSPT,BNMHT,
     *BNMAT,BNMBT,(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,AWSPT,ANMHT,ANMAT,ANMBT,
     *IDAY,BTIME,(B(I),I=1,6),(BSD(I),I=1,6),BSTME,BMTSP,BMSSP,
     *BWTSP,BWSSP,BMSPT,BWSPT,BNMHT,BNMAT,BNMBT
 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
C     APPLY THE RESISTANCE,AREA,TRANSMISSION TO
C     FIND THE MEASURED SOLAR IRRADIANCE FROM THE EARTH
C     VIEWING CHANNEL
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(1),D(1),E(1)
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
115   FORMAT(F8.7,2X,F8.7,2X,F8.7)
30    CONTINUE
C
C
C
C
      DO 40 I=1,4
      F(I) = E(I)/AREA(I)
      PRINT 116,F(1)
116   FORMAT (F7.2)
40    CONTINUE
C
C
C    CALCULATE THE ACTUAL IRRADIANCE FROM THE SOLAR
C    MONITOR CHANNEL
C
C
      F(5) = E(5) + E(6)
      F(5) = F(5)/AREA(5)
      PRINT 117,F(5)
117   FORMAT (F7.2)
C
C
C     APPLY TRANSMISSION CORRECTION ON THE SW CHANNELS
C
C
      F(2) = F(2)/SWTRN(1)
      F(4) = F(4)/SWTRN(2)
      PRINT 118,F(2),F(4)
118   FORMAT (F7.2,2X,F7.2)
      GO TO (120,130,120)IDF
C
C     CALL TO SUBROUTINE NOA9 TO DO SOME SPECIFIC PROCESSING
C     FOR TIROS
C
120   CALL NOA9(F,SPMOD,IDF,ITAPE)
      GO TO 150
C
C     CALL TO ROUTINE ERBS TO DO SOME SPECIFIC PROCESSING
C     FOR ERBS
C
130   CALL ERBS(F,SPMOD,ITAPE)
```

```
      C
      C     CALCULATE THE JULIAN DAY
      C      GET THE SUN DISTANCE USING AN ALGORITHM
      C
      C
      C
      150    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
      C
      C     JULIAN DAY REFERRED TO 1/1/2000 AS 2451545
      C
             N=JDAY-2451545
             G=357.528+0.9856003*FLOAT(N)
             SNDIST=1.00014-0.01671*COSD(G)-0.00014*COSD(2*G)
             AUCOR=1/(SNDIST)**2
      C
      C      CORRECTION FOR ONE ASTRONOMICAL UNIT
      C
             DO 60 I=1,5
             F(I) =F(I)/AUCOR
             PRINT 118,F(1),F(5)
      60     CONTINUE
      C
      C      APPLY  SUN ANGLE CORRECTION ON EV CHANNELS
      C
             DO 65 I=1,4
             F(I) = F(I)/COSD(SNANG(1))
             PRINT 118,F(1),F(4)
      65     CONTINUE
      C
      C      APPLY SUN ANGLE CORRECTION ON SMA CHANNEL
      C
             F(5) = F(5)/COSD(SNANG(2))
             PRINT 119,F(5)
      119    FORMAT (F7.2)
             DO 70 I=1,4
             PRODCT(I) = AREA(I) * RESTN(I)
      70     CONTINUE
             PRODCT(5) = AREA(5) * RESTN(6)
             PRODCT(2) = PRODCT(2) * SWTRN(1)
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
                PRODCT(4) = PRODCT(4) * SWTRN(2)
      C
      C         CALL THE HEADR ROUTINE TO WRITE THE MAIN HEADR
      C
                CALL HEADR(ITAPE,SPMOD,PRODCT,IREC)
      C
      C         WRITE THE SOLAR FLUX VALUES OF THE EARTH CH TO
      C         THE OUTPUT FILE
      C
                WRITE (ITAPE,2800)ADATE,ATIME,A(1),ASD(1),AMTSP,
               *A(2),ASD(2),AMSSP,A(3),ASD(3),AWTSP,A(4),ASD(4),AWSSP
      2800      FORMAT(1X,A8,2X,A8,5X,F6.1,2X,F4.1,4X,F4.1,5X,F6.1,2X,F4.1,4X,
               *F4.1,3X,F6.1,2X,F4.1,4X,F4.1,6X,F6.1,3X,F4.1,5X,F4.1)
                IF(B(1).LT.25.)GO TO 102
                WRITE (ITAPE,2900)IDAY,BTIME,B(1),BSD(1),BMTSP,B(2),BSD(2),
               *BMSSP,B(3),BSD(3),BWTSP,B(4),BSD(4),BWSSP
                GO TO 103
      102       WRITE(ITAPE,2901)IDAY,BTIME,BSD(1),BMTSP,B(2),BSD(2),
               *BMSSP,B(3),BSD(3),BWTSP,B(4),BSD(4),BWSSP
      2900      FORMAT (3X,I3,5X,A8,5X,F6.1,2X,F4.1,4X,F4.1,5X,F6.1,2X,
               *F4.1,4X,F4.1,3X,F6.1,2X,F4.1,4X,F4.1,6X,F6.1,3X,F4.1,5X,F4.1)
      2901      FORMAT (3X,I3,5X,A8,3X,'OFF SCALE',1X,F4.1,4X,F4.1,5X,F6.1,2X,
               *F4.1,4X,F4.1,3X,F6.1,2X,F4.1,4X,F4.1,6X,F6.1,3X,F4.1,5X,F4.1)
      103       WRITE (ITAPE,3000)IND,(F(I),I=1,4)
      3000      FORMAT(2X,I4,18X,F6.1,19X,F6.1,17X,F6.1,20X,F6.1)
      C
      C         CALL TO SUBROUTINE HEDR2 TO WRITE THE SMA MEASUREMENTS
      C
                CALL HEDR2(ITAPE)
                WRITE (ITAPE,3100)BSTME,A(5),ASD(5),A(6),ASD(6),AMSPT,
               *AWSPT,ANMHT,ANMAT,ANMBT,SNDIST,AUCOR,SNANG(1)
      3100      FORMAT(3X,A8,1X,'C',1X,F6.1,1X,F4.1,1X,'C',1X,F6.1,1X,F4.1,
               *2X,F4.1,2X,F4.1,2X,F4.1,1X,F4.1,1X,F4.1,4X,F7.5,
               *9X,F7.5,14X,F4.2,1X,'EARTH')
                WRITE (ITAPE,3200)B(5),BSD(5),B(6),BSD(6),BMSPT,BWSPT,BNMHT,BNMAT,
               *BNMBT,SNANG(2)
      3200      FORMAT(12X,'O',1X,F6.1,1X,F4.1,1X,'O',1X,F6.1,1X,F4.1,2X,F4.1,
               *2X,F4.1,2X,F4.1,1X,F4.1,1X,F4.1,41X,F4.2,1X,'SOLAR')
                WRITE (ITAPE,3300)F(5)
      3300      FORMAT(28X,F6.1)
                IF(IREC.GT.2)THEN
   1            IREC=0
```

```
1             ENDIF
              GO TO 1
      99      WRITE(6,4000)IOS
      4000    FORMAT(' ERROR ENCOUNTERED= ',I6)
      4010    RETURN
              END
```

--VARIABLE MAP--(LO=A)

-NAME	---ADDRESS	--BLOCK	-----PROPERTIES	-------TYPE	---------SIZE
A	761B			REAL	6
ADATE	1050B			CHAR*8	
AMSPT	2B	/TVAR/		REAL	
AMSSP	1060B			REAL	
AMTSP	0B	/TVAR/		REAL	
ANMAT	1063B			REAL	
ANMBT	1064B			REAL	
ANMHT	1062B			REAL	
ANS	NONE		UNUSED/*S*	CHAR*1	
AREA	6B	/COEFF/		REAL	5
ASD	1024B			REAL	6
ATIME	1051B			CHAR*8	
AUCOR	1100B			REAL	
AWSPT	6B	/TVAR/		REAL	
AWSSP	1061B			REAL	
AWTSP	4B	/TVAR/		REAL	
B	767B			REAL	6
BMSPT	3B	/TVAR/		REAL	
BMSSP	1066B			REAL	
BMTSP	1B	/TVAR/		REAL	
BNMAT	1071B			REAL	
BNMBT	1072B			REAL	
BNMHT	1070B			REAL	
BSD	1032B			REAL	6
BSTME	1053B			CHAR*8	

-NAME	---ADDRESS	--BLOCK	-----PROPERTIES	-------TYPE	---------SIZE
BTIME	1052B			CHAR*8	
BWSPT	7B	/TVAR/		REAL	
BWSSP	1067B			REAL	
BWTSP	5B	/TVAR/		REAL	
C	775B			REAL	6
D	1003B			REAL	6
E	1011B			REAL	6
F	1017B			REAL	5
G	1076B			REAL	
I	1057B			INTEGER	
IDAY	1065B			INTEGER	
IDF	1		DUMMY-ARG	INTEGER	
IND	1056B			INTEGER	
IOS	1055B			INTEGER	
IREC	1054B			INTEGER	
ITAPE	1073B			INTEGER	
JDAY	1074B			INTEGER	
N	1075B			INTEGER	
PRODCT	1042B			REAL	5
RESTN	0B	/COEFF/		REAL	6
SNANG	1040B			REAL	[illegible]
SNDIST	1077B			REAL	
SPMOD	1047B			CHAR*7	
SWTRN	13B	/COEFF/		REAL	[illegible]

ORIGINAL PAGE IS
OF POOR QUALITY

```
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.

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

ORIGINAL PAGE IS
OF POOR QUALITY

--VARIABLE MAP--(LO=A)

-NAME-	--ADDRESS	--BLOCK-	-----PROPERTIES-	-------TYPE-	--------SIZE
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	1	DUMMY-ARG		REAL	5
IDF	3	DUMMY-ARG		INTEGER	
ITAPE	4	DUMMY-ARG		INTEGER	
PPFNM	130B			REAL	
PPFNW	125B			REAL	
PPSNM	127B			REAL	

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.

```
          SUBROUTINE ERBS(F,SPMOD,ITAPE)
          COMMON/TVAR/AMTSP,BMTSP,AMSPT,BMSPT,AWTSP,BWTSP,AWSPT,BWSPT
          CHARACTER SPMOD*7
          DIMENSION F(5)
          PPFEW = 0.21
          PPSEW = 5.04
          PPFEM = 0.88
          PPSEM = 2.45
  130     SPMOD = 'ERBS '
          ITAPE = 5
          F(1) = F(1) + (AMTSP-BMTSP)*PPFEM
          F(1) = F(1) + (AMSPT-BMSPT)*PPSEM
          F(3) = F(3) + (AWTSP-BWTSP)*PPFEW
          F(3) = F(3) + (AWSPT-BWSPT)*PPSEW
          PRINT 118,F(1),F(3)
  118     FORMAT(F7.2,2X,F7.2)
          RETURN
          END
```

--VARIABLE MAP--(LO=A)

-NAME-	--ADDRESS	--BLOCK-	-----PROPERTIES-	------TYPE-	--------SIZE
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	

-NAME-	--ADDRESS	--BLOCK-	-----PROPERTIES-	------TYPE-	--------SIZE
F	1		DUMMY-ARG	REAL	5
ITAPE	3		DUMMY-ARG	INTEGER	
PPFEM	76B			REAL	
PPFEW	74B			REAL	
PPSEM	77B			REAL	
PPSEW	75B			REAL	
SPMOD	2		DUMMY-ARG	CHAR*7	

ORIGINAL PAGE IS
OF POOR QUALITY

```
      SUBROUTINE HEADR(ITAPE,SPMOD,PRODCT,IREC)
C
C
C     THIS ROUTINE PRINTS THE MAIN HEADR ,HEADR FOR THE
C     EARTH VIEWING CHANNEL
C
      COMMON/COEFF/RESTN(6),AREA(5),SWTRN(2)
      DIMENSION PRODCT(5)
      IREC=IREC+1
      IF(IREC.EQ.1 .OR. IREC .GT.3)THEN
      WRITE(ITAPE,3)
3     FORMAT(1H1,49X,'ERBE   NS   SOLAR     CALIBRATIONS')
      WRITE(ITAPE,5)
5     FORMAT (2X,'INPUT DATA')
      WRITE(ITAPE,10)
10    FORMAT (20X,'MED-TOT',14X,'MED-SW',12X,'WIDE-TOT',11X,'WIDE-SW',
     *12X,'SMA')
      WRITE(ITAPE,15)(AREA(I),I=1,5)
15    FORMAT(5X,'AREA',11X,4(F10.8,9X),F10.8)
      WRITE(ITAPE,20)(SWTRN(I),I=1,2)
20    FORMAT(5X,'TRANSMISSION',3X,'1.0',16X,F5.3,14X,'1.0',16X,F5.3)
      WRITE(ITAPE,25)(RESTN(I),I=1,4),RESTN(6)
25    FORMAT(5X,'RESISTANCE',2X,F6.1,13X,F6.1,13X,F6.1,13X,
     *F6.1,13X,F6.1)
      WRITE(ITAPE,30)(PRODCT(I),I=1,5)
30    FORMAT(5X,'PRODUCT',7X,F7.4,12X,F7.4,12X,F7.4,12X,F7.4,12X,F7.4)
      ENDIF
      WRITE(ITAPE,1000)SPMOD
1000  FORMAT('0',1X,A7)
      WRITE (ITAPE,2000)
2000  FORMAT(2X,'DATE',6X,'TIME',7X,'MEDIUM TOTAL CHANNEL',7X,
     *'MEDIUM  SW CHANNEL',7X,'WIDE TOTAL CHANNEL',7X,'WIDE SW CHANNEL')
      WRITE (ITAPE,3000)
3000  FORMAT(24X,'COUNTS',3X,'SD',4X,'NMFTLT',4X,'COUNTS',3X,'SD',
     *4X,'NMFSLT',2X,'COUNTS',3X,'SD',4X,'NWFTLT',5X,'COUNTS',4X,'SD',
     *5X,'NWFSLT')
      RETURN
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
SUBROUTINE HEDR2       74/860  OPT=1,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642          87/05/20. 15.09.32          PAGE      1
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.

                  SUBROUTINE HEDR2(ITAPE)
            C
            C
            C     THIS ROUTINE PRINTS THE SOLAR MONITOR CHANNEL HEADRS
            C
                  WRITE(ITAPE,1000)
            1000  FORMAT(2X,'TIME',8X,'SOLAR MONITOR CHANNEL')
                  WRITE (ITAPE,2000)
            2000  FORMAT(14X,'SPACE',3X,'SD',4X,'SUN',5X,'SD',3X,'MSPT',2X,
                 *'WSPT',4X,'SMA TEMPS',5X,'SUN DISTANCE',4X,
                 *'1 AU CORRECTION',7X,'SUN ANGLE')
                  RETURN
                  END


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

  ITAPE        1    DUMMY-ARG                  INTEGER


--STATEMENT LABELS--(LO=A)
 -LABEL-ADDRESS-----PROPEPTIES----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 =   25280
  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.
9. Performing Organization Name and Address ST Systems Corporation (STX) 28 Research Drive Hampton, VA 23666		10. Work Unit No. 665-45-30-01
		11. Contract or Grant No. NAS1-17851
12. Sponsoring Agency Name and Address NASA Langley Research Center Hampton, VA 23665		13. Type of Report and Period Covered Contractor Report
		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))	18. Distribution Statement
ERBE ERBS NOAA-9 NOAA-10	Unclassified - Unlimited Subject Category 61

19. Security Classif.(of this report)	20. Security Classif.(of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	91	A05

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