UNIVERSITY OF CALIFORNIA

LICK OBSERVATORY TECHNICAL REPORTS

No. 15

THE IBM-360 SCANNER DATA REDUCTION SYSTEM

S. A. HAWLEY AND K. H. NORDSIECK

Santa Cruz, California
February 1976
Amended April 1977

This is roughly the numbering system in the PDP-8 system B flux directory, and differs from the System A directory.

9. *DUMP: Dumps out all numbers for disk scan number INP(M) or scratch scan/table number - INP(M).
10. *PLOT: Causes a crude one-page line printer plot of disk scan number INP(M) (= 1 to 32) or slit buffer number -INP (= -1 to -4) to be printed out (up to 15 plots with one command). The ordinate is 50 divisions between the minimum and maximum values found in the scan; 16 channels are averaged to obtain 128 abscissa points.
11. *STORE: Stores the scan from slit buffer M (= 1 to 4) as disk scan number INP(M) (= 1 to 32).
12. *WRITE: Stores disk scan number INP on the output tape in a form acceptable to the PDP-8 SDRS system (up to 15 scans stored in order per card). A reduced data tape log is printed out including tape scan number, disk scan number, scale factor, low and high wavelengths, star name and normalizing scan name. An end-of-file is written only at the end of the job. The line printer is advanced to a new page when a *WRITE command is executed. A scan must be divided by quartz or by a standard star in order to be written properly; if the *NORM flag is not set the scale factor will not be found to convert the numbers to 6-place integers.

12. *END: Stop!

III. USING THE IBM-360 SDRS PROGRAM

This sections describes the setup required to run the IBM-360 SDRS program. There is a lot of JCL and there are probably a number of different ways to try to run this job, but the one outlined here works.

```
//SDRS JOB          (Acct. #, time limit in minutes, line limit in
                     1000's), your name, MSGLEVEL = 1
```

Column 16

Takes about 30^{m} cpu time to do 1 night's worth thru *BFLUX and *WRITE

UNIVERSITY OF CALIFORNIA

LICK OBSERVATORY TECHNICAL REPORTS

No. 15

THE IBM-360 SCANNER DATA REDUCTION SYSTEM

S. A. HAWLEY AND K. H. NORDSIECK

Santa Cruz, California
February 1976
Amended April 1977

Note on Updates to IBM 360-SDRS:

Several new commands are now available as part of the IBM 360-SDRS thanks mostly to Alan Koski. Generally the new commands combine a number of the old commands by stringing together several operations which were previously specified individually. This makes for a lot less keypunching and at a glance you can see what you've done to your data. However, since the details of the reduction procedure are now suppressed, it is necessary to understand the old commands if you want to really know what's going on. A brief description of each of the new commands has been provided by Alan Koski and should be inserted as pages 8 - 8f. Note that in addition the old *WRITE command has been modified.

There is also now a 32K Focal version of IBM 360-SDRS and the PDP8 DEC ← → IBM routines. These were written by Steve Grandi. There is no difference to concern the casual user but now you should be able to save factors of 2-3 in I/O requests. It takes 2 records to write a 2048-CH scan and 3 to write a 4096-CH scan instead of 5 and 9 as before.

Steve Hawley
April 11, 1977

TABLE OF CONTENTS

I. INTRODUCTION

The following is a description of the IBM-360 version of the Scanner Data Reduction System. Through a series of 13 commands the 360/40 will perform many of the reduction chores now assigned to the PDP-8. It is still necessary to produce the lambda calibration coefficients as before. However, the 360-SDRS program will:

a. read and sum raw data scans from tape
b. read λ-calibration coefficients
c. subtract sky and correct for atmospheric extinction
d. scrunch
e. add or divide scans
f. system A flux calibration (standard star should be summed from raw data tape and line-zapped as before on the PDP-8)
g. system B flux calibration
h. plot scans on the line printer
i. write results on reduced data tape.

Different parts of the program are accessed by commands on punched cards. The data itself may be located in any one of five places depending on the reduction stage.

1. Input raw data tape: There are programs for the PDP-8 (e.g. Jack's Dec→IBM and 9-Track Tape Handler programs on the "IBM↔DEC" tape in the cabinet in the microphotometer room--see Appendix 1) which will write a 4096 channel raw data scan (or a 2048 reduced data scan) from DECtape as a nine 512-word record scan on IBM tape. The first record is an ID record containing dwell time, hour angle, slit code, name, right ascension, declination, time and comments. These are stored on IBM tape in the order they are stored on DECtape (see J. A. Baldwin write-up in the Lick Observatory Technical Report No. 2). The subsequent 8 records contain the right and left slit spectra respectively.

2. Reduced data tape: The IBM 360/40 can write reduced data scans on 9-track tape which are in turn transferred to DEC-tape by Jack's IBM→DEC routine on the special tape in the cabinet in the microphotometer room. (See Appendix 1). The reduced data tape contains 2048 channel scans in the same format as on the raw data tape with an updated ID record, so there are five 512 word records per 2048 channel scan. The reduced data tape is assigned unit number 10 by the program and different file numbers may be assigned by appropriate JCL cards to different subsets of reduced data. WARNING -- writing a file causes succeeding files to be wiped out, so unless you want to write over old data be sure to specify succeeding file numbers on each succeeding job.
3. Internal buffers: Intermediate results are stored in six 2048-word buffers. Two are "scratch" buffers located in core; these are inaccessible to the outside user. Four are "slit" buffers located on "scratch" disk (see below); these are buffers 1-4 referred to in the commands. In dealing with raw data, they are used to accumulate spectra of star (right slit, left slit) and sky (right slit, left slit) in that order. Otherwise they may be used to process up to four 2048-channel scans in parallel by one command. The buffers are cleared at the beginning of a job. They are also automatically cleared before execution of a *READ or *ADD (add spectra into buffer from disk) command if a "clear buffer" flag has been set by previous use of the other commands (the flag is then reset after the clear). Thus a sequence of *READ and *ADD commands accumulates data in the buffers, and then the use of any other command lets the computer know that you are finished accumulating, so that the buffer will be automatically cleared just before the next accumulation sequence. The line printer output goes to a new page each time the buffer is cleared.
4. Scratch disk file: A FORTRAN direct access disk file is used automatically by the program to store temporary tables for the

duration of the job (no cost to the user!) A JCL statement assigns this unit number 3. There is room for four slit buffers (2048 channel scans 1-4), four scrunch tables (2048 word tables number 5-8), two extinction tables (9, 10) and 15 response tables. The tables are calculated only when necessary, according to scan ID information stored by code within the appropriate commands *SCRUNCH, *CORRECT, and *AFLUX, respectively.

5. Output disk file: The user may specify a semi-permanent disk file for storage of final or intermediate results. This is particularly useful if one wants to write a separate program to further process the data. The format is the same as the scratch file, except a "retention period" is specified (see below). Storage cost is about $.50 per scan per month. The unit number specified within the program is 4 and presently there is allowed room for 32 scans, although this is easily changed by modifying the FORTRAN "DEFINE FILE" statement for unit 4. All disk scans or tables have 128 words appended at the end for label information. For scans the first 64 words are ID information from the scanner and the second 64 words are used within the program.

II. THE COMMANDS

All the commands have the same card format: an asterisk (*) in the first column followed by 19 alphameric characters, and 15 integer options "INP" in I4 format. A few commands cause the program to expect additional data on cards in a different format, but with these exceptions all input cards must be in the command format. Only the asterisk and the first two characters of the command word are read and checked against the allowed commands; the remaining characters in columns 4-20 are available for identification. The contents of the input card are immediately printed out in the same format, to help describe what processing has been done. The meaning of the options varies with the command, as described below. A general rule to be remembered is that the command processes the options in the order of their appearance on the card.

1. *READ: Reads scans from raw data tape, corrects for dead time, and adds the results into the correct slit buffer according to the slit code in the ID-record. The first option, INP(1), is related to the file on tape that is to be read. As mentioned above, the user must assign a unique "unit" number to each file that he expects to use through a JCL statement (see below). Adding 100 to INP(1) will indicate that all input scans in this file are one slit (2048-word) scans: the dead time correction is not applied, and the scan is added to slit buffer number 1. In this way, a reduced data tape may be read back in. The string of scan numbers is terminated by either a -1 or the end of a card. The 2048-channel scans are weighted by 1/N where N is the number of scans read in. So if 3 scans are read into buffer number 1, each will be multiplied by 1/3.
2. *ADD: Adds scan from the output disk file into the slit buffers. Disk scan number INP(M) is added into slit buffer number M (M = 1-4). If INP(M) = 0 (or a blank) nothing is added into this buffer; if it is negative between -1 and -4 the contents of the slit buffer M - buffer number INP(M) itself are added into buffer M. The same buffer clearing rules apply as for the *READ command; an *ADD command following anything other than a *READ or another *ADD command causes the buffers to be cleared as the first step in the command; thus combining the buffers may call for temporary storage of the buffers on disk and then adding them back in. An isolated *ADD command with no options acts as a "clear buffer" command. The total dwell and scale factor are updated in the ID area. The scans are weighted in the same manner as in *READ.
3. *LAMBDA: Tells the computer to read in up to four sets of wavelength coefficients to be associated with the four calibration tables. INP(1) through INP(4) (= 1-4; blanks and other numbers are ignored) specify which calibration(s) are being read in and in what order. Each calibration occupies one card: the first 20 columns may be any identification information. The six

calibration coefficients are then entered in 6F10.0 format in the remaining space, in exactly the same form as printed out by the PDP-8 λ - calibration program.

4. *CORRECT: Performs sky and extinction corrections on summed raw data using the extinction information stored in the ID area. First the right slit (option INP(1)), then the left slit (INP(2)) is processed. The sky spectrum is first subtracted from the "star" spectrum (the possibility of different dwells as in nebular work is allowed for). This step is however skipped if INP(M) has a prefixed minus sign. The extinction correction is applied using wavelength calibration number INP(M) = 1 to 4 or -1 to -4, otherwise this step is skipped.
5. *NORM: Divides the scan in slit buffer M (= 1 to 4) by disk scan number INP(M) (≥1; otherwise no action for slit M). Zeros in the normalizing scan are treated as infinity. A flag is set indicating that this is a normalized scan.
6. *SCRUNCH: Puts the scan in slit buffer M (=1 to 4) on a linear wavelength scale using lambda calibration number INP(M). Every time a new scrunch table is calculated, a message giving the beginning wavelength and dispersion is printed out. The scan is then resampled using this table (just as in the PDP-8 system) and stored back in the original slit buffer. This command will handle either normalized or unnormalized scans. The beginning and ending wavelength and sampling interval are stored in the ID area.

   An important feature of this system which is different from the PDP-8 system is the automatic wavelength justification: the wavelength of the first channel is chosen using calibration number one (it is the largest multiple of the dispersion smaller than the wavelength of the first channel), and this is retained for other calibrations. This first channel wavelength, and the automatic 1.25Å/ch sampling interval, may be overridden

by prefixing the calibration coefficient number with a minus sign: the computer then expects a card containing the new dispersion and the new beginning wavelength in floating point format in columns 21-30 and 31-40. If the dispersion field is left blank 1.25Å/ch will be assumed; if the λ_o field is left blank the data will be scrunched as above. The first 20 columns are available for identification. Be sure to include one such card for each slit. These cards must follow the *SCRUNCH command card and preceed any other command card. The advantage of this system is that scans scrunched with different calibrations end up on one wavelength scale, and can be combined with just an *ADD command.

7. *AFLUX: Multiplies scrunched scans by stellar fluxes or by inverse - response functions. Up to 15 different curves are kept in scratch tables 11-16. Flux tables for eight stars used in system A flux calibration are available:

   1 = BD+33 2642
   2 = BD+28 4211
   3 = Feige 34
   4 = Feige 25
   5 = Feige 15
   6 = Feige 56
   7 = HILTNER 102
   8 = BD+40 4032

   The scan in slit buffer M (= 1 to 4) is divided by curve number INP(M).

8. *BFLUX: Multiplies scrunched scans by response curves created from observations of standard stars. The "BFLUX command must always be preceded by an *ADD command with the disk scan numbers of the scrunched standard star scan as INP(M), M = 1,4. In other words, a response curve can be created using from one to four standard stars which should be placed in the slit buffers beginning with buffer 1. The *BFLUX command card then contains as INP(M), M = 1,4 the code number for each standard star so that the program can get the correct set of Hayes points from core.

The program then expects a card following the *BFLUX command containing the disk scan numbers of the scrunched scans to be fluxed. As in the PDP-8 batch flux program the fluxed scans are stored back on themselves.

The program ignores any Hayes point that falls within the first six or last six angstroms of the standard star scan. It also automatically flattens the response curve if values become greater than 100 times the minimum value. When more than one standard star scan is used the final curve is produced from a straight mean of the response values at each Hayes point for all the standards. After all the data has been fluxed, the response curve is placed in slit buffer number 1 and may then be saved with *STORE command as a scan on disk and subsequently on tape.

A response curve which has been saved on tape may be read back in as a 2048-channel scan. To flux more data with this response curve the *ADD command should specify the response curve disk scan number as INP(1), the *BFLUX card should be blank, and then follow with the card of data scans as before. A flag is set indicating that the scan has been fluxed.

A line printer plot of the response curve may be obtained by following the "data scan" card with a *PLOT command specifying INP(1) = -1.

The coding for the standard stars is:

1 = Feige 34
2 = BD + 33 2642
3 = HZ 15
4 = Feige 15
5 = BD + 8 2015
6 = Feige 25
7 = Feige 56
8 = HILTNER 102
9 = BD + 25 3941
10 = BD + 40 4032
11 = BD + 28 4211

This is roughly the numbering system in the PDP-8 system B flux directory, and differs from the System A directory.

9. *DUMP: Dumps out all numbers for disk scan number INP(M) or scratch scan/table number - INP(M).

10. *PLOT: Causes a crude one-page line printer plot of disk scan number INP(M) (= 1 to 32) or slit buffer number - INP (= -1 to -4) to be printed out (up to 15 plots with one command). The ordinate is 50 divisions between the minimum and maximum values found in the scan; 16 channels are averaged to obtain 128 abscissa points.

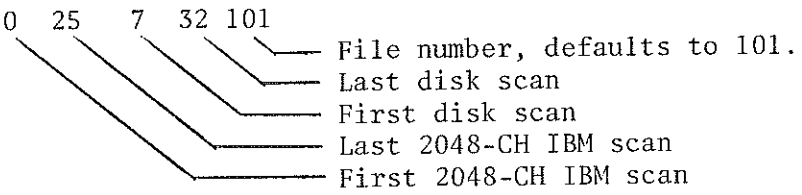
11. *STORE: Stores the scan from slit buffer M (= 1 to 4) as disk scan number INP(M) (= 1 to 32).

12. *WRITE: Stores disk scan number INP on the output tape in a form acceptable to the PDP-8 SDRS system (up to 15 scans stored in order per card). A reduced data tape log is printed out including tape scan number, disk scan number, scale factor, low and high wavelengths, star name and normalizing scan name. An end-of-file is written only at the end of the job. The line printer is advanced to a new page when a *WRITE command is executed. A scan must be divided by quartz or by a standard star in order to be written properly; if the *NORM flag is not set the scale factor will not be found to convert the number to 6-place integers.

```
*WRITE          -1  -2  -4  -3
                output onto tape from buffer (rather than disk)
                scans - INP(M)
```

13. *END: Stop!

14. *INPUT



*INPUT reads 2048-CH scans from IBM tape and stores the scans on disk.

This card *INPUT 1 6 5 10

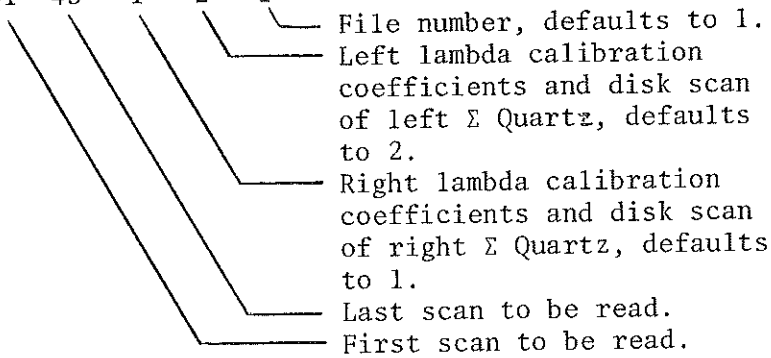
replaces these

```
*READ      101   1  -1
*STORE       5
*READ      101   2  -1
*STORE       6
*READ      101   3  -1
*STORE       7
*READ      101   4  -1
*STORE       8
```

```
*READ      101   5  -1
*STORE       9
*READ      101   6  -1
*STORE      10
```

The last disk scan number, INP(4), is not used and need not be given. As many tape scans as are specified are written beginning at the first disk scan. The last disk scan number probably should be given though so the operator will know where the data is on disk.

15. *NEON 41 43 1 2 1



*NEON reads scans from tape, does dead time corrections, divides by Σ Quartz and scrunches, but does not subtract sky or do extinction corrections (thus different from *PROCESS).

These cards

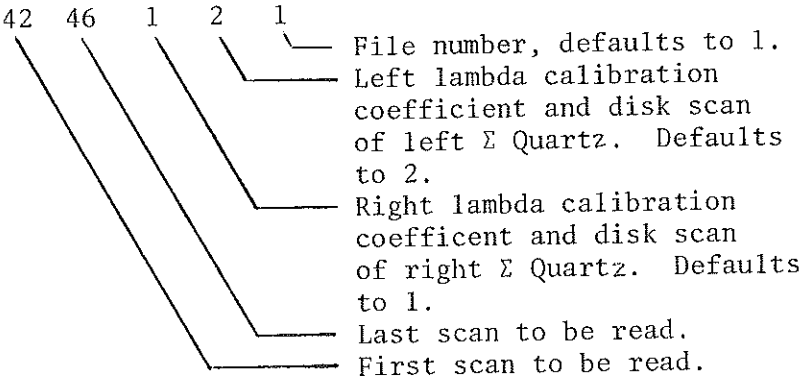
```
*NEON      41  43
*STORE     15  16
```

replace these

```
*READ       1  41  42  43  -1
*NORM       1   2
*SCRUNCH    1   2
*STORE     15  16
```

A maximum of 14 scans to be read in can be implied by *NEON. In the example above, *NEON 41 43 implies 3 scans; 41, 42, 43.

16. *PROCESS 42 46 1 2 1

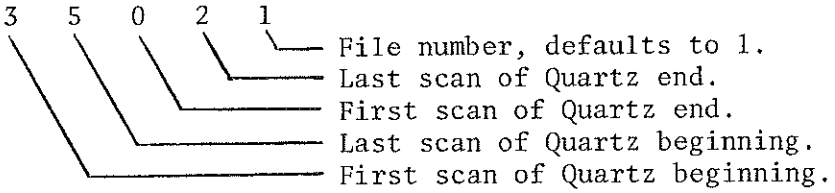


*PROCESS reads scans from tape, does dead time correction, subtracts sky, does extinction correction, divides by Σ Quartz, and scrunches.

These cards	*PROCESS	42	46					
	*STORE	13	14					
replace these	*READ	1	42	43	44	45	46	-1
	*CORRECT	1	2					
	*NORM	1	2					
	*SCRUNCH	1	2					
	*STORE	13	14					

A maximum of 14 scans to be read can be implied by *PROCESS. In the example above, *PROCESS 42 46 implies 5 scans: 42, 43, 44, 45, 46.

17. *QUARTZ



*QUARTZ reads in Quartz end, Quartz beginning and leaves Σ Quartz beginning + end in buffers 1 and 2 and Σ Quartz beginning ÷ end in buffers 3 and 4.

These cards	*QUARTZ	3	5	0	2	
	*STORE	1	2	3	4	
replace these	*READ	1	0	1	2	-1
	*STORE	33	34			
	*READ	1	3	4	5	-1
	*STORE	1	2			
	*NORM	33	34			
	*STORE	3	4			
	*ADD	1	2			
	*ADD	33	34			
	*STORE	1	2			

*QUARTZ uses DISK scans 33 and 34.

*QUARTZ needs these JCL and Fortran cards:

```
//GO.FT04F001 DD DSN=&SCRATCH,UNIT=2314,
//   SPACE=(4352,(68,2)),VOL=SER=42
DEFINE FILE 4(68,1088,U,KSCR)
```

18. *RFLUX 2 5 6
↑ ↑ ↑
identifies the standard stars (up to 4 of them).

(next card) 9 13 11
↑ ↑ ↑
locations on disk of right slits of the above identified standard stars. The left slits must be in the next scans, e.g., in 10, 14, and 12.

(next card) 13 15 17 21 25 23
↑ ↑ ↑ ↑ ↑ ↑

These give the disk scans (up to 15 of them) of the right slits to be fluxed. The left slits must be in the next scans, e.g., in 14, 16, 18, etc.

*RFLUX divides the object right slit by the standard star right slit and similarly for the left slit. The division is then system A fluxed as with *AFLUX and the fluxed slits are output onto tape. Each object will be fluxed by each standard star. The scans on disk may have been previously system B fluxed if desired. The scans on disk remain unchanged from *RFLUX.

These cards

```
*RFLUX      1   3   4
            9  11  15
           11  13  15  17  21  23
```

replace these [This is too horrible to contemplate, but I'll try.]

```
*ADD       11  12
*NORM       9  10
*AFLUX      1   1

*WRITE     -1  -2
*ADD       11  12
*NORM      11  12
*AFLUX      3   3

*WRITE     -1  -2
*ADD       11  12
*NORM      15  16
*AFLUX      4   4

*WRITE     -1  -2
*ADD       13  14
*NORM       9  10
*AFLUX      1   1

*WRITE     -1  -2
*ADD       13  14
*NORM      11  12
*AFLUX      3   3

*WRITE     -1  -2
*ADD       13  14
*NORM      15  16
*AFLUX      4   4

*WRITE     -1  -2
*ADD       15  16
etc., etc., etc., . . . . whew!
```

19. *SFLUX 2 5
↑ ↑
identifies which standard stars (up to 4 of them) are in the Buffers.

As with *BFLUX, another card must follow giving the DISK scans to be fluxed, and the response curve is returned in Buffer 1. *SFLUX finds the average RSCV, plots it, and saves it. Then each individual RSCV is found, divided by the average RSCV, and plotted. The average RSCV is used to flux the scans on DISK and it is then left in Buffer 1.

These cards
```
*ADD          9  11
*SFLUX        2   5
              9  11  13  15
*STORE        5
```

replace these
```
*ADD          9  11
*BFLUX        2   5
          blank card
*PLOT        -1
*STORE        5
*ADD          9
*BFLUX        2
          blank card
*NORM         5
*PLOT        -1
*ADD         11
*BFLUX        5
          blank card
*NORM         5
*PLOT        -1
 etc. for more standard stars
*ADD          5   rest of card is blank
*BFLUX        9  11  13  15  ← this card follows *BFLUX card.
```

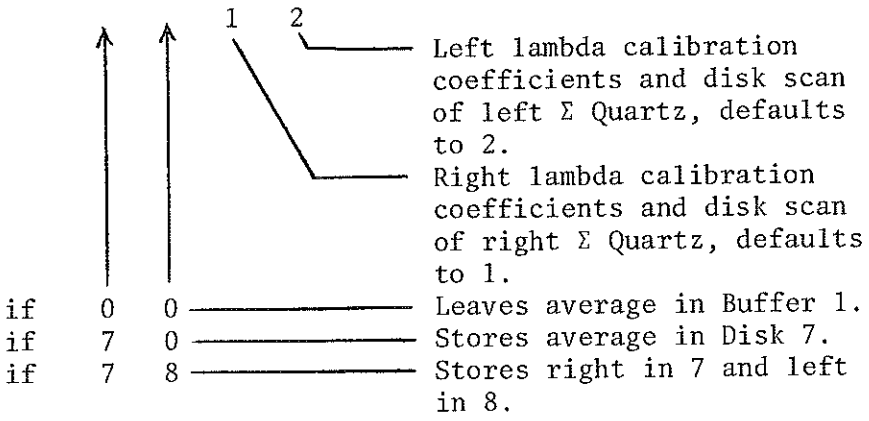
If only 1 standard star is used with *SFLUX, only the RSCV will be plotted, no divisions. If the RSCV is added into the buffer and *SFLUX is blank, the response curve is used to flux scans as when using *BFLUX and is not plotted. *ZERO may preceed *SFLUX as with *BFLUX.

*SFLUX uses DISK scan 34.

*SFLUX needs these JCL and Fortran cards:

```
//GO.FT04F001 DD DSN=&SCRATCH,UNIT=2314,
//   SPACE=(4352,(68,2)),VOL=SER=42
DEFINE FILE 4(68,1088,U,KSCR)
```

20. *SKY



*SKY divides the sky scans by Σ Quartz, scrunches them (and averages them together). The result is in Buffer 1 (and 2) and may be stored on Disk.

These cards

```
*PROCESS    42  46
*STORE      13  14
*SKY
*STORE      15
```

replace these

```
*PROCESS    42  46
*STORE      13  14
*NORM        0   0   1   2
*SCRUNCH     0   0   1   2
*STORE       0   0  33  34
*ADD        33
*ADD        34
*STORE      15
```

*SKY should follow a command to save the contents of buffer 1 since the output of *SKY appears in buffer 1 (and 2). If the sky is stored on disk with the *SKY command (see below), it also remains in the Buffer.

Note that an even more compact way is:

```
*PROCESS    42  46
*STORE      13  14
*SKY        15
```

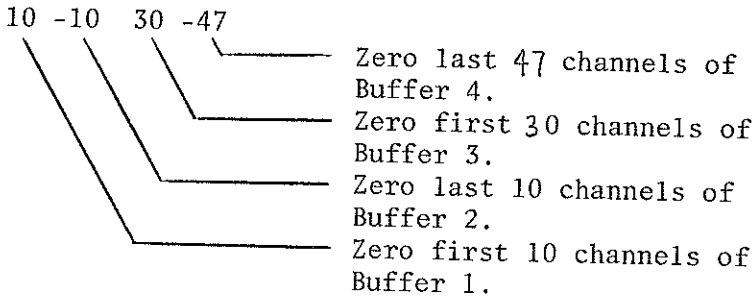
and the average is stored as Disk scan 15 (it is still in Buffer 1).

To keep the sky slits separate,

```
*PROCESS    42  46
*STORE      13  14
*SKY        15  16
```

and the right slit sky is stored in Disk scan 15 and the left slit sky is stored in Disk scan 16.

21. *ZERO



*ZERO sets channel contents to zero. If number of channels is $n > 0$, then n channels of data beginning at left end of scan are zeroed. If $n < 0$, then n channels of data beginning at right end of scan are zeroed.

*ZERO is recommended to zero bad, noisy data in the first channels of a blue scan of a standard star before fluxing by *BFLUX or *SFLUX. You want to remove all the data from Hayes points likely to be affected by the bad, noisy data. This in effect removes these Hayes points from the response curve.

Use:

```
*ADD      9  11
*ZERO    20  20
*SFLUX    2   5
          9  11  13  15
*STORE    5
```

To remove all noise from a scan use *ZERO 2048 not *ZERO -2048 because the number field is I4.

III. USING THE IBM-360 SDRS PROGRAM

This section describes the setup required to run the IMB-360 SDRS program. There is a lot of JCL and there are probably a number of different ways to try to run this job, but the one outlined here works.

```
//SDRS JOB          (Acct. #, time limit in minutes, line limit in
                     1000's), your name, MSGLEVEL = 1
```

Column 16

Takes about 30^{m} cpu time to do 1 night's worth thru *BFLUX and *WRITE

```
/*MESSAGE             PLEASE MOUNT TAPE (Raw data tape #) (UNLABELLED)
                          ↑
               Column 16
/*MESSAGE             NO WRITE RING
                          ↑
               Column 16
/*MESSAGE             PLEASE MOUNT TAPE (Reduced data tape #) (UNLABELLED)
                          ↑
               Column 16
/*MESSAGE             WRITE RING IN
                          ↑
               Column 16
```

"UNLABELLED" refers to tape format NL: the operator knows to mount "UNLABELLED" tapes on particular tape drives. The /*MESSAGE cards come immediately after the job card.

```
//   EXEC   FORTGLG
//LKED.SYSIN    DD    *
                     (OBJECT DECK)
//GO.FT03F001    DD     DSN = &DISK, UNIT = 2314, SPACE = (4352, (64, 2)),
VOL = SER = 41
```

This card sets up the scratch file for tables (unit no. 3).
There is no need to change anything on this card.

```
//GO.FT04F001    DD    DSN = &SCRATCH, UNIT = 2314, SPACE = (4352, (64, 2)),
VOL = SER = 41
```

This card establishes disk space for spectra which will be written on tape at the end of the job. Data on unit 3 or unit 4 are lost at the completion of the job.

```
//GO.FT10F001    DD LABEL = (file number, NL,, OUT),
//    VOL = SER = Reduced data tape name, DSN = Reduced data tape name
//    UNIT = 2400, DISP = (NEW, PASS), DCB = (RECFM = VS, LRECL = 2052,
BLKSIZE = 2056)
```

These three cards are necessary for writing a reduced data tape. They specify which file is to be written and on what tape. The reduced data tape is unit number 10.

If a reduced data tape is not required for the job then replace the three cards above with

```
//GO.FT10F001    DD    DUMMY
                        & pull the appropriate/*MESSAGE CARDS
//GO.FT11F001    DD    LABEL = (File number, NL,, IN),
//    VOL = SER = Raw data tape name, DSN = Raw data tape name
//    UNIT = 2400, DISP = (OLD, PASS), DCB = RECFM = US, LRECL = 2052,
BLKSIZE = 2056)
```

These cards are required for the raw data tape. The unit number for the raw data tape is the file number on the tape which you wish to read plus 10. So to read file number 1

```
//GO.FT11F001 DD LABEL = (1, NL,, IN),
//GO.SYSIN    DD    *
                        (COMMAND CARDS)
/*
```

The following is a listing of the complete JCL from a successful run. For details regarding the meanings of the abbreviations and parameters please consult the computer center publications.

User-generated JCL cards are enclosed in brackets.

```
//SDRS JOB      (    ,60,5),HAWLEY,MSGLEVEL=1                                  JOB  46
***MESSAGE     PLEASE MOUNT TAPE SC0408          (UNLABELLED)
***MESSAGE     NO WRITE RING
***MESSAGE     PLEASE MOUNT TAPE SC0407          (UNLABELLED)
***MESSAGE     WRITE RING IN
// EXEC FORTGLG
XXLKED EXEC PGM=IEWL,REGION=96K,PARM=(XREF,LET,LIST)                          00020000
XXSYSLIB   DD  DSNAME=SYS1.FORTLIB,DISP=SHR                                   00040000
XX  DD  DSN=SYS1.PLOTLIB,DISP=OLD
XXSYSLMOD  DD  DSNAME=&GOSET(MAIN),DISP=(NEW,PASS),UNIT=SYSDA,                X00060000
XX             SPACE=(1024,(20,10,1),RLSE),DCB=BLKSIZE=1024                   00080014
XXSYSPRINT DD  SYSOUT=A                                                       00100000
XXSYSUT1   DD  UNIT=SYSDA,SPACE=(1024,(100,10),RLSE),DCB=BLKSIZE=1024,        X00110018
XX             DSNAME=&SYSUT1                                                 00120018
XXSYSLIN   DD  DDNAME=SYSIN                                                   00140000
//LKED.SYSIN DD *

   ** SDRS       00.02.28  01/27/76

IEF236I ALLOC. FOR SDRS      LKED
IEF237I 235    ALLOCATED TO SYSLIB
IEF237I 235    ALLOCATED TO
IEF237I 231    ALLOCATED TO SYSLMOD
IEF237I 0F1    ALLOCATED TO SYSPRINT
IEF237I 235    ALLOCATED TO SYSUT1
IEF237I 0A2    ALLOCATED TO SYSLIN
IEF142I - STEP WAS EXECUTED - COND CODE 0000
IEF285I    SYS1.FORTLIB                                        KEPT
IEF285I    VOL SER NOS= 32     .
IEF285I    SYS1.PLOTLIB                                        KEPT
IEF285I    VOL SER NOS= 32     .
IEF285I    SYS76026.T080651.RF000.SDRS.GOSET                   PASSED
IEF285I    VOL SER NOS= 12     .
IEF285I    SYS76026.T080651.RF000.SDRS.SYSUT1                  DELETED
IEF285I    VOL SER NOS= 32     .

   ** SDRS       00.03.33  01/27/76

XXGO   EXEC  PGM=*.LKED.SYSLMOD,COND=(4,LT,LKED)                              00160000
XXFT05F001 DD  DDNAME=SYSIN                                                   00180000
XXFT06F001 DD  SYSOUT=A                                                       00200000
XXFT07F001 DD  SYSOUT=B                                                       00220000
//GO.FT03F001 DD DSN=&DISK,UNIT=2314,SPACE=(4352,(64,2)),VOL=SER=41
//GO.FT04F001 DD DSN=&SCRATCH,UNIT=2314,SPACE=(4352,(64,2)),VOL=SER=41
//GO.FT10F001 DD LABEL=(,NL),
//  VOL=SER=SC0407,DSN=SC0407,
//  UNIT=2400,DISP=(NEW,PASS),DCB=(RECFM=VS,LRECL=2052,BLKSIZE=2056)
//GO.FT11F001 DD LABEL=((1),NL,,TN),
//  VOL=SER=SC0408,DSN=SC0408,
//  UNIT=2400,DISP=(OLD,PASS),DCB=(RECFM=VS,LRECL=2052,BLKSIZE=2056)
//GO.SYSIN DD *
//SYSIN      DD *                    GENERATED STATEMENT
//

   ** SDRS       00.03.35  01/27/76

IEF236I ALLOC. FOR SDRS      GO
IEF237I 231    ALLOCATED TO PGM=*.DD
IEF237I 0A3    ALLOCATED TO FT05F001
IEF237I 0F1    ALLOCATED TO FT06F001
IEF237I 0B1    ALLOCATED TO FT07F001
```

Each user of the SDRS program will develop his own style but we include a sample night's worth of data reduction for illustrations. The first card causes scans 0, 1, and 2 to be read from input tape file 1. The sum is then stored on disk as scans 7 (right slit) and 8 (left slit). For this example this would be the sum of the quartz scans for the end of the night. Then the sum of the quartz at the beginning of the night (scans 3, 4, and 5) are read and stored on disk as scans 5 (right slit) and 6 (left slit). Next the *NORM command causes the contents of buffer 1 to be divided by disk scan 7 and the contents of buffer 2 to be divided by disk scan 8. This would be the sum of the quartz (beginning) divided by the sum of the quartz (end). The division is stored as scans 3 and 4. The *ADD command then clears the buffers and adds in disk scans 5 and 7 to buffer 1 and 6 and 8 to buffer 2. The *STORE command saves the sum of the quartz (beginning + end) as disk scans 1 (right slit) and 2 (left slit).

The quartz division is then read back into the buffers and the lambda calibration coefficients are loaded with the *LAMBDA command (right slit calibration = #1; left slit calibration = #2). The quartz division is scrunched using these coefficients and resaved on disk as scans 5 and 6.

Raw data scans are finally read in -- scans 12, 13, 14, 15, and 16 from file number 1 -- and extinction correction is applied using lambda calibration coefficients 1 on right slit and 2 on left slit. Sky subtraction is performed here as well. The *NORM card then causes the right slit to be divided by the sum of the quartz in disk scan 1 and the left slit by disk scan 2. The data is then scrunched using calibration 1 for right slit and 2 for left slit. Finally the scrunched scans are stored on disk -- scan 13 is right slit, scan 14 is left slit. The procedure is repeated for new raw data scans and finally disk scans 5 thru 25 are written on reduced data tape. NOTE: The blue data is not fluxed.

The red data is reduced in the same manner, but here the data is to be fluxed. The standard stars (right slits) are stored as disk scans 7, 9, 11, and are put in buffer numbers 1-3. Suppose they are BD + 28 4211, HZ 15, and Feige 34. Then *BFLUX will have the corresponding code numbers: INP(1) = 11, INP(2) = 3, INP(3) = 1. Scans 7, 9, 11, 13, 15, 17, 19, and 21 are fluxed and resaved on disk. The left slits are processed similarly. Finally, note that the response curves are saved as disk scans 22 and 23.

```
*READ                    1    0    1    2   -1
*STORE                   7    8
*READ                    1    3    4    5   -1
*STORE                   5    6
*NORM                    7    8
*STORE                   3    4
*ADD                     5    6
*ADD                     7    8
*STORE                   1    2
*ADD                     3    4
*LAMBDA                  1    2
RT 08/XI/75 BLUE      3445083     1924439     2356395     -893473
LF 08/XI/75 BLUE      3446359     1915264     2368681     -889229
*SCRUNCH                 1    2
*STORE                   5    6
*READ                    1    6    7   -1
*CORRECT                 1    2
*NORM                    1    2
*SCRUNCH                 1    2
*STORE                   7    8
*READ                    1    8    9   -1
*CORRECT                 1    2
*NORM                    1    2
*SCRUNCH                 1    2
*STORE                   9   10
*READ                    1   10   11   -1
*CORRECT                 1    2
*NORM                    1    2
*SCRUNCH                 1    2
*STORE                  11   12
*READ                    1   12   13   14   15   16   -1
*CORRECT                 1    2
*NORM                    1    2
*SCRUNCH                 1    2
*STORE                  13   14
*READ                    1   17   18   19   20   21   22   23   -1
*CORRECT                 1    2
*NORM                    1    2
*SCRUNCH                 1    2
*STORE                  15   16
*READ                    1   24   25   26   27   28   29   30   -1
*CORRECT                 1    2
*NORM                    1    2
*SCRUNCH                 1    2
*STORE                  17   18
*READ                    1   31   32   33   34   35   36   37   -1
*CORRECT                 1    2
*NORM                    1    2
*SCRUNCH                 1    2
*STORE                  19   20
*READ                    1   38   39   40   41   42   -1
*CORRECT                 1    2
*NORM                    1    2
*SCRUNCH                 1    2
*STORE                  21   22
*READ                    1   43   44   45   46   47   48   49   -1
*CORRECT                 1    2
*NORM                    1    2
*SCRUNCH                 1    2
*STORE                  23   24
*WRITE                   5    6    7    8    9   10   11   12   13   14   15   16   17   1
```

```
*WRITE                  19   20   21   22   23   24   25
*ADD                     3    4
*LAMBDA                  1    2
RT 08/XI/75 RED       4756478      1937546      2364081      -899839
LF 08/XI/75 RED       4757528      1930796      2368314      -893444
*SCRUNCH                 1    2
*STORE                   5    6
*READ                    1   50   51   52   -1
*CORRECT                 1    2
*NORM                    1    2
*SCRUNCH                 1    2
*STORE                   7    8
*READ                    1   53   54   55   -1
*CORRECT                 1    2
*NORM                    1    2
*SCRUNCH                 1    2
*STORE                   9   10
*READ                    1   56   57   58   -1
*CORRECT                 1    2
*NORM                    1    2
*SCRUNCH                 1    2
*STORE                  11   12
*READ                    1   59   60   61   62   63   64   65   -1
*CORRECT                 1    2
*NORM                    1    2
*SCRUNCH                 1    2
*STORE                  13   14
*READ                    1   66   67   68   69   70   71   72   -1
*CORRECT                 1    2
*NORM                    1    2
*SCRUNCH                 1    2
*STORE                  15   16
*READ                    1   73   74   75   76   77   78   79   -1
*CORRECT                 1    2
*NORM                    1    2
*SCRUNCH                 1    2
*STORE                  17   18
*READ                    1   80   81   82   83   84   -1
*CORRECT                 1    2
*NORM                    1    2
*SCRUNCH                 1    2
*STORE                  19   20
*ADD                     7    9   11
*BFLUX                  11    3    1
                         7    9   11   13   15   17   19   21
*STORE                  22
*ADD                     8   10   12
*BFLUX                  11    3    1
                         8   10   12   14   16   18   20   22
*STORE                  23
*WRITE                   5    6    7    8    9   10   11   12   13   14   15   16   17   18
*WRITE                  19   20   21
*END
```

IV. ERROR MESSAGES

The 360-SDRS program comes equipped with error messages.

1. ILLEGAL COMMAND FOR SDRS: The program requested but did not find an input card in the command format: asterisk in the first column followed by the first two characters of an accepted command. Action: halt execution.
2. END OF FILE (READ ERROR) ENCOUNTERED IN FILE nn.:
   *READ: problem encountered in reading raw data tape. File option (= unit # -10) = nn. Action: halt execution.
3. CAN'T FIND CORRECT ID FOR SCAN mm FILE nn:
   *READ: The first 64 words in an ID scan mm are not equal to the sequential number of the scan in file option nn. Tape is incorrectly written. Action: halt execution.
4. BAD WAVELENGTH INFORMATION FOR SLIT BUFFER NO. mm:
   *SCRUNCH: Wavelength coefficients for lambda calibration are bad or have not been read in.
   *AFLUX: Wavelength information for this scan is missing (probably not scrunched) so interpolated response tables cannot be set up. Action: go to text slit.

V. HELPFUL HINTS

It costs about $15-$20 to reduce a night's worth of data if SDRS is run while cheap rates apply. The cost in I/O requests (about $1 per 1000) can be reduced by copying the raw data from Dectape to IBM tape in the order in which it is to be read.

It is probably a good idea to write out the scrunched (but not fluxed) scans onto the output tape before *BFLUX is executed in case of failure. There is no provision for accepting or rejecting the response curve other than to take whatever is generated. If the curve should look strange for some reason, say one standard were bad, the scrunched data would then be available to flux by hand (by PDP-8) or to have another try on the IBM-360 with a different number of standard stars. In this case the reduction through scrunch would not have to be repeated.

When more than one standard star is used the shape of the final, average response curve is normally the only check on the fluxing. Alan Koski has suggested a method of checking on the individual response curves. Create a curve for each standard and save them as disk scans. Then create the average curve, divide each individual curve by the average, and plot the results on the line printer. The wiggles in the curve will indicate the deviation from a pure grey shift and the plot scale will indicate the amount of the grey shift.

Occassionally an error like "END OF DATA SET" or "END OF FILE" will occur caused by the operator mounting the output tape on the input tape unit. This can happen because the tapes are unlabelled. The procedure is then to run the job over and if it works the second time the computer center will credit your account for the first run.

The general philosophy behind the IBM-360 SDRS program has been to make much of the data reduction as automatic as possible. Therefore, there are no extraordinary fix-up routines incorporated into the system. If was felt that if something looks wrong when done the most straightforward way it's probably worth having a look at on the PDP-8.

Work is in progress to develop an automatic lambda calibration routine, and a Calcomp plotting routine for the IBM-360 SDRS program. The 360-SDRS program is only now starting to be used routinely on real data. There are undoubtably several bugs still undiscovered. Please report all disasters so that they may be corrected.

We are grateful to Jack Baldwin for his understanding of how the computer center works and to Alan Koski for helping to develop the *BFLUX routine, and for several suggestions leading to improvements in the program.

APPENDIX 1

Included here are the listings for the PDP-8 routines to transfer scans from Dectape to IBM tape and back. Programs 31 and 32, "DEC to IBM", were written by K. Nordsieck and modified by A. Koski. Program 34, "IBM to DEC", was written by A. Koski. Program 35, "TAPE HANDLER", was written by J. Baldwin.

```
 W
C:LICK FOCAL SCN75-C  K$E0]SF,BAX CALL(1,1);CP 32

01.01 X CALL(1,1);CP 32

02.10 S J=J+1;I (NJ-J)2.9;I (DT(J)-DT(J-1))2.2,2.4
02.20 X SWIT(-1);X STAT(100,%00,2);I (J-2)2.3;X MTAK(0,-320,1,7)
02.30 D 9;T !"MOUNT DT#"DT(J),", HIT 'ALT MODE'";D 9;A K;X STAT(-1);S D%
02.40 I (S1(J))2.1;I (S2(J))2.1;F S=S1(J),S2(J);D 3
02.50 G 2.1
02.90 D 9;X EOF(0);S TL=TL+1;D 9;X CALL(31,2)

03.10 S SS=SS+1;X MTAK(16,%*S,2,7);X CLER(1);X ISOK(23,#S-1)
03.20 X SAV(1,1);X PUTN(%,0,SS,65);F K=1,#S;D 4
03.30 T !SS,"  "S,"    ";I (-LN)3.4;I (FTAK(3,68)-27)3.9
03.40 X NAME(73);X PULL(1);X UNPK(6,2117);S B=FCOIY(12,2117);X NAME(72)
03.50 I (LN)3.9;F K=1,B;T " "
03.60 T DT(J),"   "DA,MO,YR;S LN=-1
03.90 R

04.10 I (-PWAIT(%*K))4.2;R
04.20 X BAK(1);X [EMP(0);G 4.1

09.10 X STAT(-1);S K1=100;D 10;D 10;S K1=30;D 10;D 10;X STAT(0,0,2)

10.50 T "";F K=0,K1;S K=K
*
```

```
C:LICK FOCAL SCN75-C  LAE0

01.01 X CALL(1,1);C-PG 34  IBM TO DEC

02.10 T !!"IBM TO DEC: 2048-CH SCANS WITH ID"!!
02.20 X NAME(72);X RWND(0);S LS=-1
02.30 A !"GET IBM SCANS: 1ST,LAST"S1;I (S1)1.01;A S2;I (S2)2.3
02.40 A " STORE AS DEC(7) SCANS"S3;I (S3)2.3;S S4=S3+S2-S1
02.50 T %2," TO",S4;I (S4-36)3.1;T " > 35",!;G 2.4

03.10 S DI=S1-LS-1;I (DI)3.3,4.1;X ADV(5*DI)
03.20 G 4.1
03.30 S DI=-DI;I (DI-LS/2)3.4;S LS=-1;X RWND(0);G 3.1
03.40 X BAK(5*DI)

04.10 F J=S1,S2;D 5
04.20 S LS=S2;G 2.3

05.10 F K=1,5;S D=FREAD(3*K);D 6
05.20 X PULL(1,1);X ISOR(23,4,1)
05.30 X MPUT(16,32*(S3+J-S1),32,7)

06.10 I (D)6.2,6.3;R
06.20 T !%2"IBM TAPE ERROR, RECORD",K," SCAN" J;R
06.30 T !%2"END OF FILE ENCOUNTERED ON IBM TAPE"
06.40 T !"WHEN TRYING TO READ RECORD",K," SCAN"J
06.50 A !"CONTINUE?"O;I (O-MY)6.6;S K=K-1;R
06.60 S J=S2;S K=5;X RWND(0);S S2=-1
*
```

```
 X CALL(35)
*W
C:LICK FOCAL SCN75-C  N!M&

01.01 X CALL(1,1);C-PG 35
01.10 C-9-TRACK TAPE HANDLER + DEC TO IBM TRANSFER FOR 2048 CH SCANS.

08.05 X SWI1(-1);X NAME(72)
08.10 X STAT(-1);T !!!"9 TRACK TAPE HANDLER"!!!;X STAT(300,700,1)
08.20 T "OPT"!" 1  REWND"!" 2  IBME"!" 3  EOF"!" 4  FIND EOF"
08.30 T !" 5  COUNT TO EOF"!" 6  ADVANCE N RECORDS"
08.40 T !" 7  BACK UP N RECORES"!" 8  WRITE"!" 9  READ";X STAT(-1)
08.50 A !"OPT"J;I (FITR((J-1)/9))8.5,8.6,8.5
08.60 X GO(9,10*J)

09.10 X REWND(0);G 8.5
09.20 X IBME(0);G 8.5
09.30 X EOF(0);G 8.5
09.40 X ADV(4000);G 8.5
09.50 S N=0
09.52 S N=N+1;S D=FREAD(0,0,0,0,0,1);I (D)9.56,9.54,9.52
09.54 T "FOUND "%4,N," RECORDS, = "%6.02,N/5,N/9," SCANS";G 8.5
09.56 T "TAPE ERROR, RECORD "%4,N;G 9.52
09.60 A "N"N;X ADV(N,1);G 8.5
09.70 A "N"N;X BAK(N,1);G 8.5
09.80 G 10.1
09.90 G 11.1

10.10 A "DEC SCAN"S;I (S)8.5;A "  UNIT"U;I (U)8.5;X MTAK(16,32*S,32,U)
10.20 X CLEAR(1);X ISOR(23,4);X SAV(1,1);X PUTN(3,0,S,64)
10.30 F J=1,5;I (-FSAIT(3*J))10.5
10.40 G 8.5
10.50 T !"9-TRACK TAPE ERROR!"

11.10 A "  TO DEC SCAN"S;I (S)8.5;A "  UNIT"U;I (U)8.5
11.20 F J=1,5;I (FREAD(3*J))10.5
11.30 X PULL(1,1);X ISOR(23,4,1);X MPUT(16,32*S,32,U);G 8.5
*
```

APPENDIX 2

The following is a listing of the SDRS program as of February 25, 1976.

```
C      PROGRAM SDRS                                                          IBM00-01
C      PROGRAM TO READ RAW I.T.S. DATA OFF TAPE , CORRECT FOR PAIRED
C       PULSES AND EXTINCTION , ADD SCANS , SUBTRACT SKY , NORMALIZE ,
C       CALIBRATE INTENSITY AND WAVELENGTH , AND STORE ON DISK .
C      IBM 360-40 VERSION  -  9 - TRACK TAPES                                IBM01-01
C      --- K.NORDSIECK ,  FEB, 1974 .                                        IBM01-02
C      MODIFIED AND UPDATED--S.A. HAWLEY--AUGUST, 1975
C      SYSTEM B FLUX CALIBRATION--AUGUST 1975--S.A. HAWLEY
       DIMENSION B1I(2176),B2I(2176),IBUF(512),ID1(64),ID2(64)
       DIMENSION INP(15),KTAB(10),DWT(4),COUNTS(4),DW(16),HA(16),TAB(128)
       INTEGER ALPHA(10)
       DIMENSION RSCV(2048),SUMPNT(29),PT(29),IPT(29),SPNT(29),
      X SFLUX(29,11)
       DIMENSION BF34(29),BBD33(29),BHZ15(29),BF15(29),BBD8(29),BF25(29),
      X BF56(29),BH102(29),BBD25(29),BBD40(29),BBD28(29)
       DOUBLE PRECISION COF(6,4),COFT                                        IBM02-01
       COMMON /BUFFER/ BI(2176,2)
       EQUIVALENCE (BI,IBUF),(BI,B1I),(BI(1,2),B2I)
       EQUIVALENCE (B2I(2049),XTO),(B2I(2050),DXT),(B2I(2051),FNXT),
      X  (B2I(2052),SC),(B2I(2049),TAB),(B2I(2049),WO),(B2I(2050),ALA1),
      X  (B2I(2145),DW),(B2I(2161),HA)
       EQUIVALENCE (B1I(2049),ID1),(B2I(2049),ID2)
       EQUIVALENCE (B1I(1),RSCV(1))
       EQUIVALENCE (SFLUX(1,1),BF34),(SFLUX(1,2),BBD33),(SFLUX(1,3),
      X BHZ15),(SFLUX(1,4),BF15),(SFLUX(1,5),BBD8),(SFLUX(1,6),BF25),
      X  (SFLUX(1,7),BF56),(SFLUX(1,8),BH102),(SFLUX(1,9),BBD25),(SFLUX(1,
      X10),BBD40),(SFLUX(1,11),BBD28)
       DEFINE FILE 3(64,1088,U,KSCR)
       DEFINE FILE 4(64,1088,U,KSAV)
       DATA LASTER/'*'/,LL/' L'/,LS/' S'/,LR/' R'/,LREAD/'RE'/
      X  ,LCOR/'CO'/,LAFLUX/'AF'/,LBFLUX/'BF'/,LNORM/'NO'/,LSTORE/'ST'/
      X  ,LADD/'AD'/,LLAMB/'LA'/,LSCRN/'SC'/,LWRITE/'WR'/,LPLOT/'PL'/
      X  ,LDUMP/'DU'/,LEND/'EN'/
       DATA SPNT/3200.,3250.,3300.,3350.,3400.,3450.,3500.,3571.,3636.,
      2 3704.,3862.,4036.,4167.,4255.,4464.,4566.,4765.,5000.,5263.,
      3 5556.,5840.,6056.,6436.,6790.,7100.,7550.,7780.,8090.,8370./
       DATA BF34/2.780,2.844,2.913,2.816,2.787,2.711,
      2 2.687,2.559,2.496,2.405,2.276,2.156,2.004,1.912,1.780,1.695,
      3 1.516,1.413,1.281,1.171,1.043,.9736,.8566,.7870,.7332,.6564,
      4 .6687,.6059,.6087/
       DATA BBD33/1.886,1.966,2.019,1.993,1.988,1.952,1.934,1.937,1.912,
      2 1.972,2.446,2.359,2.245,2.210,2.085,2.044,1.950,1.788,1.720,
      3 1.623,1.504,1.440,1.321,1.257,1.236,1.145,1.141,1.080,0.9745/
       DATA BHZ15/.3054,.3093,.3130,.3330,.3263,.3221,
      2 .3269,.3264,.3242,.3299,.4055,.3981,.3985,.3996,.3736,.3862,
      3 .3781,.3360,.3555,.3373,.3122,.3145,.2919,.2803,.2692,.2441,
      4 .2489,.2212,.1925/
       DATA BF15/0.7572,0.7936,0.8686,0.8598,0.8839,0.8888,0.9070,0.9264,
      2 0.9489,1.011,2.377,3.037,3.009,2.976,2.892,2.881,2.795,2.684,
      3 2.592,2.512,2.412,2.314,2.186,2.084,2.027,1.904,1.857,1.826,1.80/
       DATA BBD8/0.3093,0.3192,0.4289,0.4033,0.4769,0.4512,0.5277,0.4989,
      2 0.6059,0.6742,0.7106,1.208,1.340,1.306,1.700,1.879,2.064,2.082,
      3 2.295,2.561,2.729,2.837,3.014,3.148,3.227,3.324,3.480,3.474,
      4 3.407/
       DATA BF25/0.3432,0.3500,0.3514,0.3681,0.3984,0.3793,0.3820,0.3821,
      2 0.3763,0.3908,0.6679,0.7596,0.7410,0.7216,0.6788,0.6917,0.6312,
      3 0.6426,0.6065,0.5786,0.5571,0.5301,0.4927,0.4919,0.4891,0.4324,
      4 0.4426,0.4139,0.4367/
       DATA BF56/1.000,1.074,1.091,1.095,1.137,1.116,1.137,1.106,1.110,
      2 1.133,1.778,1.885,1.803,1.755,1.661,1.653,1.562,1.472,1.337,
      3 1.315,1.225,1.172,1.083,1.037,0.9808,0.8847,0.8257,0.8143,0.7677/
```

```
      DATA BH102/.5813,.6138,.6849,.6918,.7365,.7461,.7813,.8166,.8426,
     2 .8726,1.086,1.159,1.242,1.289,1.391,1.515,1.628,1.857,2.220,
     3 2.642,2.772,2.948,3.218,3.597,3.834,4.111,4.309,4.596,4.673/
      DATA BBD25/.5195,.5450,.5829,.6160,.6564,.6736,.7099,.7447,.7642,
     2 .8098,1.138,1.253,1.326,1.405,1.471,1.623,1.751,1.923,2.291,
     3 2.616,2.694,2.800,3.017,3.248,3.382,3.574,3.675,3.739,3.709/
      DATA BBD40/1.356,1.440,1.545,1.530,1.578,1.611,1.616,1.634,1.628,
     2 1.652,2.089,2.180,2.220,2.259,2.262,2.331,2.325,2.338,2.430,
     3 2.417,2.338,2.441,2.448,2.430,2.477,2.287,2.325,2.386,2.164/
      DATA BBD28/5.148,5.263,5.425,5.248,5.287,5.035,
     2 4.961,4.756,4.622,4.500,4.238,3.865,3.641,3.490,3.227,3.045,
     3 2.813,2.587,2.410,2.176,1.944,1.835,1.661,1.539,1.409,1.280,
     4 1.242,1.174,1.118/
      KSP(K)=(K-1)*NRC+1
      INT(K)=K-4096*(K/2048)
      KCR=5                                                          IBM05-01
      KLP=6                                                          IBM05-02
      NRC1=1
      NRC=2
      MILP= 1000000
      MILN=-1000000
      DO 3 I=1,11
      DO 2 J=1,29
      IF (SFLUX(J,I) .EQ. 0.) GO TO 2
      SFLUX(J,I)=SFLUX(J,I)*1.E-24
    2 CONTINUE
    3 CONTINUE
      DO 4 L=1,16
      I1=(L-1)*128
      DO 8 I=1,128
    8 B2I(960+I)=B1I(I1+I)
    4 WRITE(3'KSP(10+L)+NRC1) (B2I(I), I=1,1088)
      DEAD=2.097E-04
      DEGRAD=57.2958
      KSCAN=-1
      KFILE=1
      KUNIT=11
      KUNOT=10
      KOUT=-1
      DO 5 K=1,10
    5 KTAB(K)=0
      DLA=0.
      RLA0=0.
   10 KBUF=-1
   15 READ(KCR,1000) ALPHA,INP
   20 CONTINUE
      IF (ALPHA(1).NE.LASTER) GO TO 970
      IF (ALPHA(2).EQ.LREAD) GO TO 100
      IF (ALPHA(2).EQ.LADD) GO TO 200
      WRITE (KLP,1001) ALPHA,INP
      IF (ALPHA(2).EQ.LLAMB) GO TO 250
      IF (ALPHA(2).EQ.LCOR) GO TO 300
      IF (ALPHA(2).EQ.LNORM) GO TO 400
      IF (ALPHA(2).EQ.LSCRN) GO TO 500
      IF(ALPHA(2) .EQ. LAFLUX) GO TO 600
      IF (ALPHA(2) .EQ. LBFLUX) GO TO 650
      IF (ALPHA(2).EQ.LSTORE) GO TO 800
      IF (ALPHA(2).EQ.LWRITE) GO TO 850
      IF (ALPHA(2).EQ.LDUMP) GO TO 700
      IF (ALPHA(2).EQ.LPLOT) GO TO 750
      IF (ALPHA(2).NE.LEND) GO TO 970
```

```
      IF (KOUT.NE.-1) END FILE KUNOT                                    IBM06-C
      STOP
C
C     ZERO THE BUFFERS AND BUFFER TABLES
C
  100 IF (KBUF) 110,120,120
  110 WRITE (KLP,1002)
      DO 113 M=1,2
      DO 113 I=1,2176
  113 BI(I,M)=0.
      DO 114 I=1,64
      ID1(I)=0
  114 ID2(I)=0
      DO 116 M=1,4
      WRITE (3'KSP(M)) B1I
      COUNTS(M)=0.
      DWT(M)=0.
  116 KTAB(M)=0
      KBUF=0
      GO TO 20
  120 CONTINUE
      WRITE (KLP,1001) ALPHA,INP
      WRITE (KLP,1200)
      N=0
C
C     FIND FILE ON TAPE
C
  125 CONTINUE
      N=N+1
      IF (N.GE.15) GO TO 180                                            IBM09-01
      IF (INP(N).LE.0) GO TO 180                                        IBM09-02
      KFORM=9                                                           IBM09-03
      IF (INP(N).GT.100) KFORM=5                                        IBM09-04
      KFILE=MOD(INP(N),100)                                             IBM09-05
      IF (KUNIT.EQ.KFILE+10) GO TO 135                                  IBM09-06
      REWIND KUNIT                                                      IBM09-07
      KUNIT=KFILE+10
  134 KSCAN=-1
C
C     FIND NEXT SCAN ON TAPE
C
  135 CONTINUE
      N=N+1
      IF (N.GE.15) GO TO 180
      IF (INP(N).LT.0) GO TO 125
      KBUF=KBUF+1
  138 K1=KFORM*(INP(N)-KSCAN-1)
      IF (K1) 140,149,145
  140 K1=-K1
      IF (2*K1 .LE. KSCAN*KFORM) GO TO 141
      REWIND KUNIT
      KSCAN=-1
      GO TO 138
  141 DO 142 K=1,K1
  142 BACKSPACE KUNIT
      GO TO 149
  145 CONTINUE
      DO 147 K=1,K1
  147 READ (KUNIT,END=900,ERR=910) D
  149 KSCAN=INP(N)
C
```

```
C       READ ID RECORD
C
  150 READ (KUNIT,END=900,ERR=910) ID1,ID2
      IF (ID1(1).NE.INP(N)) GO TO 960
      IF (ID1(2).NE.INP(N)) GO TO 960
      DT=0.
      DDW=0.
      IF (ID2(5).EQ.4095) GO TO 152
      IF (KFORM.EQ.5) ID2(2)=ID2(2)+4096*ID2(1)
      DDW=ID2(2)
      IF (KFORM.EQ.5) GO TO 152
      DT=DEAD/DDW
      IF (ID2(5).GT.30) ID2(5)=(ID2(5)-14)/10
  152 CONTINUE
      ID1(2)=DDW
      DO 159 I=3,64
  159 ID1(I)=ID2(I)
      DO 160 I=5,11
  160 ID1(I)=LETTER(ID2(I),0)
      DO 161 I=12,16
  161 ID1(I)=INT(ID2(I))
      DO 162 I=20,32
  162 ID1(I)=LETTER(0,0)
      DO 163 I=33,64
  163 ID1(I)=LETTER(ID2(I),0)
      ID1(3)=INT(ID1(3))
      ID1(4)=INT(ID1(4))
      HHA=(15.*FLOAT(ID1(3))+.25*FLOAT(ID1(4)))/DEGRAD
      LSLIT=ID1(5)
      WRITE (KLP,1600) ID1
C
C       ADD SCAN INTO DISK BUFFER ACCORDING TO SLIT CODE
C
      MM=0
      IF (LSLIT.EQ.LL.OR.LSLIT.EQ.LS) MM=2
      IF (KFORM .EQ. 5) MM=0
      ME=2
      IF (KFORM .EQ. 5) ME=1
      DO 170 M=1,ME
      MT=M+MM
      FIND (3'KSP(MT))
      DO 165 K=1,4
      READ (KUNIT,END=900,ERR=910) IBUF                          IBM14-01
      KI=512*(K-1)
      DO 165 I=1,512                                             IBM15-01
      BT=IBUF(I)
  165 B2I(I+KI)=BT/(1.-BT*DT)
      READ (3'KSCR) B1I
      KTAB(MT)=KTAB(MT)+1
      IF (ID2(5).EQ.4095) GO TO 167
      DO 166 I=1,2048
      COUNTS(MT)=COUNTS(MT)+B2I(I)
  166 B2I(I)=B1I(I)+B2I(I)
      K=KTAB(MT)
      HA(K)=HHA
      DW(K)=DDW
      DWT(MT)=DWT(MT)+DDW
      MM=0
      IF (LSLIT.EQ.LR.OR.LSLIT.EQ.LS) MM=2
      ID2(2)=DWT(MT)
      GO TO 170
```

```
 167  CONTINUE
      WT1=FLOAT(KTAB(MT)-1)/FLOAT(KTAB(MT))
      WT2=10.**(INT(ID2(1)))/FLOAT(KTAB(MT))
      DO 168 I=1,2045
 168  B2I(I)=WT1*B1I(I)+WT2*B2I(I)
      IF (B2I(2048).LT.2.E5) GO TO 170
      DO 169 I=2047,2048
      B2I(I+67)=B2I(I)
 169  B2I(I)=0.
      B2I(2046)=0.
      LD=(B2I(2114)-B2I(2115))/(128000.)+0.5
      B2I(2116)=0.625*FLOAT(LD)
 170  WRITE (3'KSP(MT)) B2I
      GO TO 135
 180  CONTINUE
      WRITE (KLP,1800) COUNTS,DWT
      GO TO 15
C
C     ADD BUFFER SCANS -INP(M) OR DISK SCANS INP(M) INTO BUFFER M
C
 200  IF (KBUF) 110,210,210
 210  CONTINUE
      WRITE (KLP,1001) ALPHA,INP
      DO 230 M=1,4
      NN=INP(M)
      IF (NN.EQ.0) GO TO 230
      IF (NN.GT.0) READ (4'KSP(NN)) B2I
      IF (NN.LT.0) READ (3'KSP(-NN)) B2I
      READ (3'KSP(M)) B1I
      KTAB(M)=KTAB(M)+1
      IF (ID2(5).EQ.4095) GO TO 221
      DO 220 I=1,2048
 220  B2I(I)=B1I(I)+B2I(I)
      ID2(2)=ID1(2)+ID2(2)
      GO TO 229
 221  CONTINUE
      WT1=FLOAT(KTAB(M)-1)/FLOAT(KTAB(M))
      WT2=1./FLOAT(KTAB(M))
      DO 222 I=1,2048
 222  B2I(I)=WT1*B1I(I)+WT2*B2I(I)
 229  WRITE (3'KSP(M)) B2I
 230  CONTINUE
      GO TO 15
C
C     READ LAMBDA COEFFICIENTS FROM CARDS
C
 250  CONTINUE
      DO 260 N=1,4
      NN=INP(N)
      IF (NN.LE.0.OR.NN.GT.4) GO TO 260
      READ (KCR,2550) ALPHA,(COF(J,NN),J=1,6)
      WRITE (KLP,2551) ALPHA,(COF(J,NN),J=1,6)
      COF(2,NN)=COF(2,NN)/2.D0
      COF(3,NN)=COF(3,NN)/8.D0
      KTAB(4+NN)=0
      IF (KTAB(9).EQ.NN) KTAB(9)=0
      IF (KTAB(10).EQ.NN) KTAB(10)=0
      IF (NN.GT.1) GO TO 260
      DO 255 K=5,10
 255  KTAB(K)=0
      DLA=0.
```

```
      RLA0=0.
  260 CONTINUE
      GO TO 10
C
C     SUBTRACT SKY AND CORRECT FOR EXTINCTION
C
  300 CONTINUE
      DO 360 M=1,4
      NN=INP(M)
      IF (NN.LT.0) GO TO 313
      IF (M.GT.2) GO TO 314
      READ (3'KSP(M)) B1I
      READ (3'KSP(M+2)) B2I
      IF (ID2(2).EQ.0) GO TO 315
      DWRAT=FLOAT(ID2(1))/FLOAT(ID1(1))
      DO 310 I=1,2048
  310 B1I(I)=B1I(I)-B2I(I)*DWRAT
      WRITE (3'KSP(M)) B1I
  313 NN=IABS(NN)
  314 IF (NN.EQ.0.OR.NN.GT.4) GO TO 360
  315 CONTINUE
      MT=MOD(M-1,2)+1
      IF (KTAB(8+MT).EQ.NN) GO TO 325
      IF (COF(1,NN).LT.2.D3.OR.COF(1,NN).GT.1.D4) GO TO 355
      READ (3'KSP(11)) B2I
      DO 320 I=1,2048
  320 B2I(I)=I-1
      CALL POLY(COF(1,NN),6,B2I,2048)
      CALL INTERP(TAB(9),XT0,DXT,IFIX(FNXT),B2I,2048)
      WRITE (3'KSP(8+MT)) B2I
      KTAB(8+MT)=NN
  325 READ (3'KSP(8+MT)) B1I
      READ (3'KSP(M)+NRC1) (B2I(I),I=1089,2176)
      DO 335 I=1,2048
  335 B2I(I)=0.
      WT1=0.
      DO 345 K=1,16
      IF (DW(K).LE.0.) GO TO 345
      DEC=(INT(ID2(15))+
     X         FLOAT(ISIGN(INT(ID2(16)),INT(ID2(15))))/600.)/DEGRAD
      SECZ=1./(.606548*SIN(DEC)+.795047*COS(DEC)*COS(HA(K)))
      ATM=SECZ-(.0018167*(SECZ-1.)+.002875*(SECZ-1.)**2 +
     X .000883*(SECZ-1.)**3)
      FAC=.921034*ATM
      WT1=WT1+DW(K)
      DO 340 I=1,2048
  340 B2I(I)=B2I(I)+DW(K)*EXP(-FAC*B1I(I))
  345 CONTINUE
      READ (3'KSP(M)) B1I
      DO 350 I=1,2048
      IF (B2I(I).EQ.0) B2I(I)=1.E50
  350 B1I(I)=B1I(I)*WT1/B2I(I)
      WRITE (3'KSP(M)) B1I
      GO TO 360
  355 WRITE (KLP,5370) M
  360 CONTINUE
      GO TO 10
C
C     NORMALIZE
C
  400 CONTINUE
```

```
      DO 420 M=1,4
      NN=INP(M)
      IF (NN.LE.0) GO TO 420
      READ (4'KSP(NN)) B2I
      READ (3'KSP(M)) B1I
      DWRAT=FLOAT(ID2(2))/FLOAT(ID1(2))
      DO 410 I=1,2048
      IF (B2I(I).EQ.0) B2I(I)=1.E50
 410  B1I(I)=B1I(I)*DWRAT/B2I(I)
      ID1(5)=4095
      ID1(1)=0
      DO 415 L=6,11
 415  ID1(L+19)=ID2(L)
      WRITE (3'KSP(M)) B1I
 420  CONTINUE
      GO TO 10
C
C     SCRUNCH ( RE-SAMPLE ON LINEAR LAMBDA SCALE ) BUFFERS 1 - 4
C      ACCORDING TO CALIBRATION TABLE NUMBER LISTED ON CARD
C
 500  CONTINUE
      DO 547 M=1,4
      N=INP(M)
      IF (N.EQ.0.OR.IABS(N).GT.4) GO TO 547
      IF (N.GT.0) GO TO 505
      N=-N
      READ (KCR,5050) ALPHA,DLA,RLA0
      WRITE (KLP,5051) ALPHA,DLA,RLA0
      DO 504 K=5,8
 504  KTAB(K)=0
 505  CONTINUE
      IF (KTAB(4+N).NE.0) GO TO 540
      IF (DLA.EQ.0.) DLA=1.25
      IF (RLA0.NE.0.) GO TO 507
      IF (COF(1,1).LT.2.D3.OR.COF(1,1).GT.1.D4) GO TO 537
      RLA0=FLOAT(IFIX(SNGL(COF(1,1))/DLA))*DLA
 507  CONTINUE
      IF (COF(1,N).LT.2.D3.OR.COF(1,N).GT.1.D4) GO TO 537
      DO 510 I=1,2048
      B1I(I)=FLOAT(I)-.5
 510  B2I(I)=0.
      COFT=COF(1,N)
      COF(1,N)=COFT-(RLA0-.5*DLA)
      CALL POLY(COF(1,N),6,B1I,2048)
      RLA1=RLA0+FLOAT(IFIX((B1I(2048)-B1I(1))/DLA))*DLA
      COF(1,N)=COFT
      B0=COF(1,N)-.5*COF(2,N)-(RLA0-.5*DLA)
      II=AMAX1(B0/DLA+1.,0.)
      I0=0
      DO 530 I=1,2048
 521  RLA=FLOAT(II)*DLA
      IF (RLA.GT.B1I(I)) GO TO 525
      C=(RLA-B0)/(B1I(I)-B0)
      IF (I0.LE.0) GO TO 522
      B2I(II)=C-C0+FLOAT(I-I0)
      GO TO 523
 522  W0=C+FLOAT(I-1)
 523  II=II+1
      IF (II.GT.2048) GO TO 535
      C0=C
      I0=I
```

```
      GO TO 521
  525 B0=B1I(I)
  530 CONTINUE
  535 WRITE (3'KSP(4+N)) B2I
      KTAB(4+N)=1
      WRITE (KLP,5350) N,RLA0,DLA
      GO TO 545
  537 WRITE (KLP,5370) M
      GO TO 547
  540 CONTINUE
      READ (3'KSP(4+N)) B2I
  545 CONTINUE
      READ (3'KSP(M)) B1I
      IFLG=0
      IF (ID1(5).NE.4095) IFLG=-1
      CALL SCRNCH(B1I,2048,W0,B2I,2048,IFLG)
      DO 546 I=1,2048
  546 B1I(I)=B2I(I)
      ID1(32)=DLA*1000.
      B1I(2114)=RLA1*100.
      B1I(2115)=RLA0*100.
      B1I(2116)=DLA
      WRITE (3'KSP(M)) B1I
  547 CONTINUE
      GO TO 10
C
C     APPLY INTENSITY CORRECTIONS TO SCRUNCHED DATA
C
  600 CONTINUE
      DO 640 M=1,4
      N=INP(M)
      IF (N.LE.C.OR.N.GT.15) GO TO 640
      READ (3'KSP(M)) B1I
      READ (3'KSP(11+N)) B2I
      RRLA=B1I(2115)/100.
      DDLA=B1I(2116)
      IF (RRLA.EQ.B2I(2053).OR.DDLA.EQ.B2I(2054)) GO TO 620
      IF (RRLA.LT.2.E3.OR.RRLA.GT.1.E4) GO TO 645
      IF (DDLA.EQ.0.) DDLA=DLA
      IF (DDLA.EQ.0.) DDLA=1.25
      DO 605 I=1,2048
  605 B2I(I)=RRLA+DDLA*FLOAT(I-1)
      CALL INTERP(TAB(9),XT0,DXT,IFIX(FNXT),B2I,2048)
      B2I(2053)=RRLA
      B2I(2054)=DDLA
      DO 610 I=1,2048
  610 B2I(I)=SC*B2I(I)
      WRITE (3'KSP(11+N)) B2I
      WRITE (KLP,5350) N,RRLA,DDLA
  620 CONTINUE
      DO 630 I=1,2048
  630 B1I(I)=B1I(I)*B2I(I)
      ID1(2)=1000
      WRITE (3'KSP(M)) B1I
  640 CONTINUE
      GO TO 10
  645 WRITE (KLP,5370) M
      GO TO 640
C
C     SYSTEM B FLUX CALIBRATION
C
```

```
  650 CONTINUE
      JK=0
      DO 651 J=1,29
      SUMPNT(J)=0.0
  651 CONTINUE
      M=1
      K=INP(M)
      IF (K .EQ. 0) GO TO 698
C
C     READ STANDARD STAR SCAN FROM BUFFER
C
  652 READ (3'KSP(M)) B2I
      B2I(2114)=B2I(2114)*0.01
      B2I(2115)=B2I(2115)*0.01
C
C     SEARCH FOR APPROPRIATE FIRST HAYES POINT
C
      DO 653 I=1,29
      IF (B2I(2115) .LT. SPNT(I) - 5.*B2I(2116)) GO TO 655
  653 CONTINUE
  655 CONTINUE
      N=I
      L=1
  656 CONTINUE
      DISP=B2I(2116)
      IF ( N .GT. 29) GO TO 690
      SUM=0.0
      BW=50.
      IF (SPNT(N) .GT. 5263.) BW=100.
      IF (SPNT(N) - B2I(2115) .LT. BW/2.+2.*DISP) BW=(SPNT(N)-B2I(2115)
     1-2.*DISP)*2.
      IF (B2I(2114)-SPNT(N) .LT. BW/2+2.*DISP) GO TO 680
      NN=(SPNT(N)-B2I(2115))/B2I(2116) + .5
      MM=.5*BW/B2I(2116) + .5
      I1=NN-MM
      I2=I1+2*MM
      DO 662 J=I1,I2
      SUM=SUM+B2I(J)
  662 CONTINUE
      AV=SUM/(FLOAT(I2-I1)+1.)
      PT(L)=SFLUX(N,K)/AV
      IPT(L)=NN
      N=N+1
      L=L+1
      GO TO 656
  680 CONTINUE
      IF (B2I(2114) - SPNT(N) .LE. 5.*DISP) GO TO 690
      BW=(B2I(2114)-SPNT(N)-2.*DISP)*2.
      NN=(SPNT(N)-B2I(2115))/DISP + .5
      MM=.5*BW/DISP + .5
      I1=NN-MM
      I2=I1+2*MM
      DO 687 J=I1,I2
      SUM=SUM+B2I(J)
  687 CONTINUE
      AV=SUM/(FLOAT(I2-I1)+1.)
      IPT(L)=NN
      PT(L)=SFLUX(N,K)/AV
      L=L+1
      N=N+1
  690 CONTINUE
```

```
      L=L-1
      N=N-1
C
C     PT CONTAINS RESPONSE VALUES, IPT CONTAINS CHANNEL NUMBERS
C
      JJ=1
      DO 691 J=1,L
      IF (PT(J) .LT. 0.) JJ=J+1
      SUMPNT(J)=PT(J)+SUMPNT(J)
  691 CONTINUE
      IF (JJ .GT.JK) JK=JJ
      M=M+1
      K=INP(M)
      IF (K .NE. 0) GO TO 652
      M=M-1
      DO 692 J=JK,L
      PT(J)=SUMPNT(J)/FLOAT(M)
  692 CONTINUE
      CALL FIT(PT,IPT,JK,L,RSCV)
      ID1(2)=1
      ID1(5)=4095
      B1I(2114)=B2I(2114)*100.
      B1I(2115)=B2I(2115)*100.
      B1I(2116)=B2I(2116)
      DO 61 I=13,23
   61 ID1(I)=0
      ID1(12)=2
      DO 62 I=15,21,3
   62 ID1(I)=47
      DO 65 I=1,M
      K=3*I+11
      ID1(K)=MOD(INP(I),10)+48
      K=K-1
      ID1(K)=(INP(I)-MOD(INP(I),10))/10+48
   65 CONTINUE
      K=3*M+12
      DO 68 I=K,24
   68 ID1(I)=0
      DO 72 I=6,11
      J=2*I
      K=J+1
   72 ID1(I)=64*ID1(J)+ID1(K)
      DO 75 I=12,23
   75 ID1(I)=0
      NPT=L-JK+1
      WRITE (6,6940) NPT
 6940 FORMAT (///,10X,'RESPONSE CURVE CREATED:',//,10X,'HAYES POINTS',
     X I17)
      NNN=N-NPT+1
      WRITE (6,6960) SPNT(NNN),SPNT(N)
 6960 FORMAT (10X,'HAYES WAVELENGTHS',F8.0,2X,F8.0)
      WRITE (6,6970) IPT(JK),IPT(L)
 6970 FORMAT (10X,'HAYES CHANNELS',2I10)
      WRITE (6,6950) B2I(2115),B2I(2114)
 6950 FORMAT (10X,'LAMBDA LOW/HIGH',2X,2F10.2)
C
C     HIT SCRUNCHED SCANS WITH THE RESPONSE CURVE AND STORE
C     THE RESULTS BACK ON DISK
C
  695 CONTINUE
      READ (5,1000) ALPHA,INP
```

```
      WRITE (6,6980)
 6980 FORMAT (///,5X,'FLUX SCANS   LAMBDA LOW   LAMBDA HIGH   ANG/CH',
     X '   NAME           STANDARD'/)
      DO 696 J=1,15
      NN=INP(J)
      IF (NN .EQ. 0) GO TO 696
      READ(4'KSP(NN)) B2I
      DO 693 I=1,2048
      B2I(I)=B2I(I)*RSCV(I)
  693 CONTINUE
      ID2(2)=10C0
      WRITE(4'KSP(NN)) B2I
      DO 675 I=1,6
      ID2(I+11)=LETTER(ID2(I+5),0)
      ID2(I+17)=LETTER(ID2(I+24),0)
  675 CONTINUE
      WRITE (6,6990) NN,B2I(2115),B2I(2114),B2I(2116),(ID2(L),L=12,23)
 6990 FORMAT (11X,I2,1X,-2P2F13.2,0PF11.3,2(3X,6A2))
  696 CONTINUE
      WRITE(3'KSP(1)) B1I
      GO TO 10
  698 CONTINUE
      READ (3'KSP(1)) B1I
      GO TO 695
C
C     DUMP OUT NUMBERS FOR SCRATCH SCAN/TABLE (-INP) OR SAVE SCAN (INP)
C
  700 CONTINUE
      DO 720 N=1,15
      NN=INP(N)
      IF (NN) 710,720,715
  710 READ (3'KSP(-NN)) B1I
      IF (NN.GT.-5) GO TO 717
      WRITE (KLP,7100) NN,B1I
      GO TO 720
  715 READ (4'KSP(NN)) B1I
  717 WRITE (KLP,7150) NN,B1I
  720 CONTINUE
      GO TO 10
C
C     PLOT BUFFER (-INP) OR DISK SCAN (INP) ON LINE PRINTER
C
  750 CONTINUE
      DO 770 N=1,15
      NN=INP(N)
      IF (NN) 755,770,757
  755 NN=-NN
      IF (NN.GT.4) GO TO 770
      READ (3'KSP(NN)) B1I
      GO TO 760
  757 READ (4'KSP(NN)) B1I
  760 CALL LINPLT(B1I,2048,B2I)
  770 CONTINUE
      GO TO 10
C
C     STORE RESULTS ON DISK OUTPUT FILE
C
  800 CONTINUE
      DO 810 M=1,4
      NN=INP(M)
      IF (NN.LE.0) GO TO 810
```

```
      READ (3'KSP(M)) B1I
      WRITE (4'KSP(NN)) B1I
  810 CONTINUE
      GO TO 10
C
C     TRANSFER DISK OUTPUT TO TAPE
C
  850 CONTINUE
      WRITE (KLP,8400)
 8400 FORMAT (1H1)
      WRITE (KLP,8500)
      DO 890 N=1,15
      NN=INP(N)
      IF (NN.LE.0) GO TO 890
      READ (4'KSP(NN)) B1I
      IF (ID1(5).EQ.4095) GO TO 851
      NPOW=0
      IF (ID1(2).LT.4096) GO TO 853
      ID1(1)=ID1(2)/4096
      ID1(2)=MOD(ID1(2),4096)
      GO TO 853
  851 BMAX=-1.E50
      DO 852 I=1,2045
  852 IF (B1I(I).GT.BMAX) BMAX=B1I(I)
      NPOW=IFIX(ALOG10(BMAX)+50.)-55
      B1I(2113)=NPOW
      ID1(1)=2048-ISIGN(2048,ID1(1))+NPOW
  853 SC=10.**(-NPOW)
      IF (B1I(2115).EQ.0.) GO TO 855
      DO 854 I=2046,2048
  854 B1I(I)=B1I(I+67)/SC
  855 CONTINUE
      SD=SC
      DO 856 L=1,6
      ID2(L)=LETTER(ID1(L+5),0)
  856 ID2(L+6)=LETTER(ID1(L+24),0)
      KOUT=KOUT+1
      WRITE (KUNOT) KOUT,KOUT,KOUT,MILP,MILN,(KOUT,L=6,64),ID1,
     X (DUM,J=1,384)
      DO 870 K=1,4
      KI=512*(K-1)
      DO 865 I=1,512
      II=I+KI
  865 IBUF(I)=B1I(II)*SD+0.5
  870 WRITE (KUNOT) IBUF
      WRITE (KLP,8600) KOUT,NN,NPOW,B1I(2115),B1I(2114),B1I(2116),
     X (ID2(L),L=1,12)
  890 CONTINUE
      SC=SD
      GO TO 10
C
C     ERROR MESSAGES
C
  900 WRITE (KLP,9000) KFILE
      STOP
  910 WRITE (KLP,9100) KFILE
      STOP
  960 WRITE (KLP,9600) INP(N),KFILE
      STOP
  970 WRITE (KLP,9700)
      STOP
```

```
1000  FORMAT(A1,9A2,1X,15I4)
1001  FORMAT(1HC,A1,9A2,1X,15I4)
1002  FORMAT(1H1)
1200  FORMAT(///1X,'INPUT SCANS FROM TAPE :'//3X,
     X 'SCAN  DWELL      H.A.   SLIT       NAME          R.A.       DECL
     X      P.S.T.  GRAT     COMMENTS'/19X,
     X 'H   M',26X,'H  M  S      D 10M      H   M'/)
1600  FORMAT(1X,2I6,3X,2I4,A6,4X,6A2,2X,3I3,3X,2I4,3X,2I4,I7,1X,13A1/
     X 1H+,86X,16A2/87X,16A2)
1800  FORMAT(//1X,'TOTAL COUNTS',4F15.0/1X,'TOTAL DWELL ',4F15.0)
2550  FORMAT(10A2,3PF10.0,6PF10.0,9PF10.0,13PF10.0,16PF10.0,19PF10.0)
2551  FORMAT(1X,10A2,3PF10.0,6PF10.0,9PF10.0,13PF10.0,16PF10.0,19PF10.0)
5050  FORMAT(10A2,6F10.3)
5051  FORMAT(1X,10A2,6F10.3)
5350  FORMAT(10X,'TABLE NO. ',I3,':  1ST CH =',F10.4,' ANG;  SAMPLE ',
     X F10.4,' ANG/CH')
5370  FORMAT(//1X,'BAD WAVELENGTH INFORMATION FOR SLIT BUFFER NO.',I5,
     X ' -- COMMAND IGNORED')
7100  FORMAT (1H1,'DUMP OF #',I5//64(8E15.6/))
7150  FORMAT (1H1,'DUMP OF #',I5//4(64(8E15.6/)/),8(8I15/)/(8E15.6))
8500  FORMAT(///1X,'OUTPUT SCANS ONTO TAPE :'//3X,'TAPE  DISK  SCALE',
     X '   LAM(LOW)    LAM(HIGH)    ANG/CH',8X,'NAME',8X,'STANDARD'/
     X 3X,'SCAN  SCAN'/)
8600  FORMAT(1X,3I6,-2P2F12.2,0PF10.3,2(3X,6A2))
9000  FORMAT(//1X,'END OF FILE ENCOUNTERED IN FILE',I5)
9100  FORMAT(//1X,'READ ERROR ENCOUNTERED IN FILE',I5)
9600  FORMAT(//1X,'CAN"T FIND CORRECT ID FOR SCAN',I5,' , FILE',I5)
9700  FORMAT(//1X,'ILLEGAL COMMAND FOR SDRS')
      END
      BLOCK DATA
      DIMENSION XTINC(103),BD33(78),BD28(76),F34(78),F25(75),F15(78),
     X F56(78),H102(78),BD40(78),
     X POL(77),PAN(77),PAN1(59),PAN2(59)
      COMMON /BUFFER/ B1I(2176),B2I(2176)
      EQUIVALENCE (XTINC,B1I),(BD33,B1I(129)),(BD28,B1I(257)),
     X (F34,B1I(385)),(F25,B1I(513)),(F15,B1I(641)),(F56,B1I(769)),
     X (H102,B1I(897)),(BD40,B1I(1025)),
     X (PAN1,B1I(1537)),(PAN2,B1I(1665)),(POL,B1I(1793)),(PAN,B1I(1921))
      DATA XTINC/3000.,100.,95.,1.,4*0.,
     3 3.35,1.75,1.12,.910,.770,.690,.625,.565,.515,.475,
     4 .435,.400,.370,.345,.320,.300,.282,.265,.250,.240,
     5 .230,.222,.218,.213,.211,.209,.206,.203,.201,.199,
     6 .197,.185,.172,.168,.162,.158,.140,.130,.150,.160,
     7 .150,.110,.130,.120,.108,.094,.082,.070,.050,.050,
     8 .068,.070,.068,.062,.058,.054,.042,.032,.030,.032,
     9 .06,.10,.11,.11,.11,.11,.11,.10,.065,.06,25*.06/
      DATA BD33/3000.,100.,70.,1.E-24,4*0.,
     3 2.192,2.163,2.138,2.095,2.063,2.058,1.987,1.960,2.500,2.600,
     4 2.500,2.382,2.317,2.254,2.194,2.136,2.080,2.027,1.976,1.927,
     5 1.880,1.836,1.793,1.752,1.712,1.675,1.639,1.605,1.572,1.541,
     6 1.511,1.483,1.456,1.430,1.405,1.382,1.359,1.338,1.317,1.297,
     7 1.278,1.260,1.242,1.225,1.208,1.192,1.176,1.161,1.146,1.131,
     8 1.116,1.101,1.086,1.071,1.056,1.041,1.026,1.010,0.994,0.977,
     9 .9601,.9425,.9241,.9051,.8853,.8647,.8432,.8207,.7972,.7726/
      DATA BD28/3000.,100.,68.,1.E-24,4*0.,
     3 6.448,6.165,5.893,5.633,5.384,5.146,4.876,4.690,4.501,4.298,
     4 4.110,3.932,3.763,3.602,3.450,3.306,3.170,3.042,2.921,2.806,
     5 2.699,2.598,2.503,2.414,2.331,2.253,2.180,2.113,2.049,1.990,
     6 1.934,1.883,1.835,1.790,1.747,1.708,1.670,1.635,1.601,1.569,
     7 1.538,1.508,1.478,1.449,1.419,1.390,1.360,1.329,1.297,1.264,
     8 1.229,1.193,1.154,1.112,1.068,1.021,0.971,0.917,0.859,0.797,
```

```
9 .7303,.6592,.5830,.5016,.4147,.3219,.2230,.1178/
 DATA F34/3000.,100.,70.,1.E-24,4*0.,
3 3.340,3.201,3.050,2.861,2.904,2.786,2.636,2.511,2.423,2.357,
4 2.258,1.163,2.072,1.985,1.902,1.823,1.747,1.675,1.607,1.542,
5 1.480,1.421,1.366,1.313,1.263,1.216,1.172,1.130,1.091,1.054,
6 1.020,.9874,.9572,.9289,.9025,.8779,.8549,.8335,.8137,.7952,
7 .7779,.7619,.7470,.7330,.7199,.7076,.6960,.6849,.6744,.6642,
8 .6543,.6446,.6350,.6253,.6156,.6056,.5953,.5846,.5734,.5616,
9 .5490,.5357,.5214,.5061,.4897,.4720,.4531,.4328,.4109,.3874/
 DATA F25/3300.,100.,67.,1.E-24,4*0.,
3                .3514,.3984,.3820,.3771,.3700,.6800,.7500,
4 .7554,.7448,.7305,.7167,.7032,.6901,.6775,.6652,.6533,.6418,
5 .6305,.6198,.6093,.5992,.5894,.5800,.5709,.5621,.5536,.5454,
6 .5376,.5299,.5227,.5157,.5089,.5024,.4962,.4903,.4846,.4791,
7 .4739,.4689,.4641,.4596,.4552,.4511,.4471,.4434,.4398,.4364,
8 .4332,.4302,.4273,.4245,.4219,.4195,.4172,.4150,.4129,.4110,
9 .4091,.4074,.4057,.4041,.4027,.4013,.3999,.3987,.3974,.3963/
 DATA F15/3000.,100.,70.,1.E-24,4*0.,
3 .7506,.7708,.8108,.8608,.9111,.9519,.9711,1.036,2.400,3.080,
4 3.160,3.146,3.106,3.066,3.027,2.987,2.948,2.909,2.871,2.833,
5 2.795,2.757,2.720,2.684,2.647,2.611,2.575,2.540,2.505,2.470,
6 2.436,2.403,2.369,2.337,2.304,2.272,2.241,2.210,2.180,2.150,
7 2.121,2.092,2.064,2.036,2.009,1.198,1.956,1.931,1.906,1.882,
8 1.859,1.836,1.814,1.792,1.771,1.751,1.732,1.713,1.695,1.677,
9 1.661,1.645,1.630,1.616,1.602,1.589,1.577,1.566,1.556,1.546/
 DATA F56/3000.,100.,70.,1.E-24,4*0.,
3 .7425,.9198,1.032,1.094,1.120,1.125,1.122,1.160,1.800,1.950,
4 1.930,1.880,1.810,1.748,1.705,1.663,1.623,1.584,1.547,1.511,
5 1.476,1.442,1.410,1.379,1.348,1.319,1.291,1.264,1.238,1.213,
6 1.188,1.165,1.142,1.120,1.099,1.078,1.058,1.039,1.020,1.001,
7 .9832,.9657,.9485,.9316,.9151,.8987,.8826,.8667,.8509,.8351,
8 .8194,.8037,.7880,.7722,.7563,.7402,.7239,.7073,.6905,.6733,
9 .6558,.6378,.6194,.6005,.5811,.5611,.5405,.5192,.4973,.4745/
 DATA H102/3000.,100.,70.,1.E-24,4*0.,
3 .4384,.4893,.5713,.6566,.7365,.7813,.8283,.8700,1.020,1.099,
4 1.141,1.202,1.260,1.294,1.338,1.441,1.525,1.570,1.642,1.743,
5 1.857,1.994,2.132,2.291,2.455,2.585,2.681,2.742,2.768,2.817,
6 2.898,2.979,3.062,3.145,2.229,3.313,3.396,3.480,3.564,3.647,
7 3.729,3.811,3.891,3.971,4.049,4.125,4.200,4.272,4.343,4.411,
8 4.477,4.541,4.601,4.658,4.713,4.763,4.811,4.854,4.894,4.929,
9 4.961,4.988,5.010,5.027,5.040,5.047,5.049,5.045,5.035,5.020/
 DATA BD40/3000.,100.,70.,1.E-24,4*0.,
3 .5979,1.068,1.358,1.515,1.585,1.616,1.628,1.648,1.876,2.144,
4 2.169,2.193,2.216,2.237,2.258,2.277,2.296,2.313,2.329,2.344,
5 2.358,2.371,2.383,2.393,2.403,2.412,2.419,2.426,2.431,2.435,
6 2.439,2.441,2.442,2.443,2.442,2.440,2.437,2.433,2.429,2.423,
7 2.416,2.408,2.400,2.390,2.379,2.368,2.355,2.342,2.327,2.312,
8 2.296,2.279,2.260,2.241,2.222,2.201,2.179,2.156,2.133,2.109,
9 2.083,2.057,2.030,2.003,1.974,1.944,1.914,1.883,1.851,1.818/
 DATA PAN1/3000.,100.,51.,1.,4*0.,
3 4.137,3.549,3.042,2.910,3.065,3.320,3.495,3.501,3.382,3.256,
4 3.195,3.160,3.180,3.215,3.245,3.298,3.348,3.396,3.471,3.542,
5 3.623,3.717,3.827,3.956,4.080,4.196,4.318,4.427,4.500,4.583,
6 4.664,4.701,4.704,4.697,4.690,4.675,4.608,4.566,4.480,4.429,
7 4.376,4.301,4.219,4.153,4.115,4.083,4.044,3.978,3.909,3.772,
8 3.639/
 DATA PAN2/3000.,100.,51.,1.,4*0.,
3 5.048,4.234,3.565,3.327,3.428,3.643,3.765,3.722,3.520,3.347,
4 3.195,3.092,3.037,3.021,3.008,3.017,3.028,3.047,3.078,3.108,
5 3.142,3.201,3.275,3.362,3.447,3.539,3.642,3.740,3.802,3.861,
6 3.918,3.965,3.976,3.973,3.959,3.935,3.880,3.845,3.723,3.597,
```

```
     7 3.556,3.556,3.536,3.534,3.547,3.571,3.589,3.579,3.550,3.463,
     8 3.463/
      DATA POL/2800.,100.,69.,1.,4*0.,
     2                                                  6.897,6.684,
     3 6.238,5.160,4.254,3.981,4.082,4.310,4.390,4.248,3.973,3.705,
     4 3.469,3.318,3.213,3.147,3.085,3.032,2.981,2.955,2.934,2.932,
     5 2.936,2.981,3.027,3.088,3.159,3.211,3.283,3.327,3.344,3.356,
     6 3.348,3.323,3.292,3.237,3.169,3.102,3.029,2.953,2.874,2.813,
     7 2.743,2.677,2.612,2.552,2.489,2.435,2.393,2.360,2.305,2.241,
     8 2.180,2.099,2.010,1.905,1.783,1.678,1.575,1.481,1.403,1.340,
     9 1.287,1.247,1.217,1.193,1.175,1.157,1.148/
      DATA PAN/2800.,100.,69.,1.,4*0.,
     2                                                  6.061,5.821,
     3 5.453,4.560,3.818,3.556,3.647,3.851,3.946,3.852,3.611,3.390,
     4 3.195,3.055,2.966,2.897,2.839,2.796,2.755,2.727,2.718,2.714,
     5 2.717,2.740,2.774,2.821,2.868,2.909,2.953,2.984,2.989,2.992,
     6 2.994,2.986,2.950,2.912,2.870,2.834,2.774,2.730,2.666,2.622,
     7 2.578,2.525,2.468,2.421,2.391,2.364,2.333,2.287,2.240,2.154,
     8 2.071,1.982,1.866,1.761,1.647,1.548,1.462,1.395,1.332,1.277,
     9 1.242,1.214,1.192,1.166,1.153,1.143,1.139/
      END
      FUNCTION LETTER(C,D)
C   STANDARD FUNCTION FOR TRANSLATING BETWEEN TOM'S PACKED ASCII
C   AND EBCDIC PACKED IN A2 FORMAT.
C   D=0 FOR ASCII TO EBCDIC, D=1 FOR EBCDIC TO ASCII.
C
C      THE EBCDIC TO ASCII CONVERSION OPTION IS DISABLED IN THIS VERSION
C      DUE TO THE REMOVAL OF THE LARGE TABLE ASCTBL(186) TO GAIN SPACE.
C       - - K. N.   AUG 1973.
C
      INTEGER LETTER,A1,A2,A3,EBC2,EBC3,EBCTBL(64),C
      DATA EBCTBL/64,193,194,195,196,197,198,199,200,201,209,210,211,
     1 212,213,214,215,216,217,226,227,228,229,230,231,232,233,77,111,
     2 93,111,111,64,90,127,123,91,108,80,125,77,93,92,78,107,96,75,97,
     3 240,241,242,243,244,245,246,247,248,249,122,94,76,126,110,111/
C   ----ASCII TO EBCDIC----
C   UNPACK ASCII
  100 A1 = IABS(C) - (IABS(C)/4096)*4096
      A2=A1/64
      A3 = A1 - A2*64
C   CONVERT TO EBCDIC
      EBC2 = EBCTBL(A2+1)
      EBC3 = EBCTBL(A3+1)
C   PACK UP THE EBCDIC
      LETTER = EBC3*65536+16448
      IF (EBC2-128) 120,120,125
  120 LETTER = LETTER + EBC2*16777216
      RETURN
  125 LETTER = LETTER + (EBC2-128)*16777216 - 2147483647
      RETURN
      END
      SUBROUTINE LINPLT(B,NB,S)
C
C      MAKE LINE PRINTER PLOT OF ARRAY B, LENGTH NB. AVERAGE ENOUGH
C      POINTS TO FIT IN 128 ELEMENTS. S(1)-S(256) IS USED AS WORK SPACE.
C      -- K.NORDSIECK,  JULY, 1974
C
      DIMENSION B(1),S(1)
      DATA ASTER/'*'/,BLNK/' '/
      NAV=(NB+127)/128
      NPTS=NB/NAV
```

IBM27-01

```
      KLP=6
      BMAX=-1.E50
      BMIN=1.E50
      DO 100 I=1,NB
      IF (B(I).GT.BMAX) BMAX=B(I)
      IF (B(I).LT.BMIN) BMIN=B(I)
  100 CONTINUE
      SC=(BMAX-BMIN)/50.
      IF (SC .EQ. 0.) SC=1.0
      WRITE (KLP,1000) BMIN,BMAX,NAV
      DO 200 J=1,NPTS
      BT=0.
      IJ=(J-1)*NAV
      DO 150 I=1,NAV
  150 BT=BT+B(IJ+I)
  200 S(J)=IFIX((BT/FLOAT(NAV)-BMIN)/SC)
      DO 210 J=1,128
  210 S(128+J)=BLNK
      DO 300 K=1,51
      DO 250 J=1,NPTS
      BT=51-K
      S(128+J)=BLNK
  250 IF (S(J).EQ.BT) S(128+J)=ASTER
      WRITE (KLP,3000) (S(J),J=129,256)
  300 CONTINUE
 1000 FORMAT(1H1,1X,'LINE PRINTER PLOT,   MIN =',E12.5,'   MAX =',
     X E12.5,',   PLOT AVERAGE OF ',I5,' POINTS'/////)
 3000 FORMAT(1X,'-',128A1)
      RETURN
      END
      SUBROUTINE INTERP(TAB,X0,DX,NX,A,NA)
C     INTERPOLATE IN TABLE TAB WITH INITIAL ABSCISSA X0, SPACING DX,
C      NUMBER OF ENTRIES NX; ANSWERS ARE STORED OVER DESIRED ABSCISSAS
C      IN ARRAY A ,  LENGTH  NA .
C     METHOD: FOR 0 - 2 AND NX-1 TO NX+1, USE NEWTON'S FORWARD AND
C      BACKWARD FORMULAS; FOR 2 TO NX-1 USE BESSEL'S FORMULA. KEEP FIRST
C      THROUGH THIRD DIFFERENCES (CUBIC FIT).
      DIMENSION TAB(1),A(1)
      NX1=NX-1
      NX2=NX-2
      DXI=1./DX
      DO 100 N=1,NA
      X=A(N)
      A(N)=0.
      S=(X-X0)*DXI
      I=S+1
      IF (I.LT.2) GO TO 10
      IF (I.GT.NX2) GO TO 20
      S=S-FLOAT(I-1)
      D2=TAB(I+2)-TAB(I+1)
      D1=TAB(I+1)-TAB(I)
      D0=TAB(I)-TAB(I-1)
      DD2=D2-D1
      DD1=D1-D0
      A(N)=TAB(I)+S*(D1+.5*(S-1.)*(.5*(DD2+DD1)+
     X .333333*(S-.5)*(DD2-DD1)))
      GO TO 100
   10 IF (I.LT.0) GO TO 100
      D1=TAB(2)-TAB(1)
      D2=TAB(3)-TAB(2)
      DD1=D2-D1
```

```
      DDD1=(TAB(4)-TAB(3)-D2)-DD1
      A(N)=TAB(1)+S*(D1+.5*(S-1.)*(DD1+.333333*(S-2.)*DDD1))
      GO TO 100
   20 IF (I.GT.NX+1) GO TO 100
      D1=TAB(NX)-TAB(NX1)
      D2=TAB(NX1)-TAB(NX2)
      DD1=D1-D2
      DDD1=DD1-(D2-(TAB(NX2)-TAB(NX-3)))
      S=S-FLOAT(NX1)
      A(N)=TAB(NX)+S*(D1+.5*(S+1.)*(DD1+.333333*(S+2.)*DDD1))
  100 CONTINUE
      RETURN
      END
      SUBROUTINE POLY(COF,NC,X,NX)
      DOUBLE PRECISION COF(1)
      DIMENSION X(1)
      DO 10 J=1,NC
      NJ=NC-J+1
      IF (COF(NJ)) 20,10,20
   10 CONTINUE
   20 CONTINUE
      NJ1=NJ-1
      DO 30 I=1,NX
      XX=X(I)
      X(I)=COF(NJ)
      DO 30 J=1,NJ1
      NJJ=NJ1-J+1
      X(I)=DBLE(X(I)*XX)+COF(NJJ)
   30 CONTINUE
      RETURN
      END
      SUBROUTINE SCRNCH(Y,NY,W0,DW,NW,KFLG)
C     IF KFLG=-1, DON'T DIVIDE BY BIN SIZE.
C     -- K.NORDSIECK,  JULY, 1974
      DIMENSION Y(1),DW(1)
      IN=W0+1
      S0=W0-IFIX(W0)
      IF (W0.LT.0.) S0=1.+S0
      DO 100 IOUT=1,NW
      BIN=DW(IOUT)
      DW(IOUT)=0.
      IF (BIN.LE.0.) GO TO 100
      KSW=IFIX(BIN+S0)-1
      S1=BIN+S0-1.-KSW
      IF (KSW.GE.0) GO TO 10
      F=S1-S0
      GO TO 50
   10 F=1.-S0
      DW(IOUT)=DW(IOUT)+SAFE(Y,NY,IN)*F
      IF (KSW.EQ.0) GO TO 40
      DO 30 K=1,KSW
   30 DW(IOUT)=DW(IOUT)+SAFE(Y,NY,IN+K)
   40 IN=IN+KSW+1
      F=S1
   50 CONTINUE
      DW(IOUT)=DW(IOUT)+SAFE(Y,NY,IN)*F
      IF (KFLG.NE.-1) DW(IOUT)=DW(IOUT)/BIN
      S0=S1
  100 CONTINUE
      RETURN
      END
```

```
      FUNCTION SAFE(A,NA,I)
      DIMENSION A(1)
      SAFE=0.
      IF (I.LE.0.OR.I.GT.NA) RETURN
      SAFE=A(I)
      RETURN
      END
      SUBROUTINE FIT(PT,IPT,JK,L,RSCV)
      DIMENSION PT(1),IPT(1),RSCV(1)
      NPT=L-JK+1
      IF (NPT. GE. 4) GO TO 800
      X1=FLOAT(IPT(JK))
      Y1=PT(JK)
      X2=FLOAT(IPT(JK+1))
      Y2=PT(JK+1)
      X3=FLOAT(IPT(L))
      Y3=PT(L)
      A=(((Y3-Y1)/(X3-X1) -(Y2-Y1)/(X2-X1))/(X3-X2))
      B=(Y3-Y1)/(X3-X1)-A*(X3-X1)
      C=Y1-(A*X1+B)*X1
      DO 810 J=1,2048
      RSCV(J)=C+B*FLOAT(J)+A*FLOAT(J)*FLOAT(J)
  810 CONTINUE
      GO TO 50
  800 CONTINUE
      J=NPT/2+1
      Y1=PT(J-2)
      X1=FLOAT(IPT(J-2))
      Y2=PT(J-1)
      X2=FLOAT(IPT(J-1))
      Y3=PT(J)
      X3=FLOAT(IPT(J))
      Y4=PT(J+1)
      X4=FLOAT(IPT(J+1))
      A=(((Y4-Y1)/(X4-X1)-(Y4-Y2)/(X4-X2))*(1./(X1-X2))-((Y4-Y1)/(X4-X1)
     1-(Y3-Y1)/(X3-X1))*(1./(X4-X3)))*(1./(X2-X3))
      B=((Y4-Y1)/(X4-X1)-(Y4-Y2)/(X4-X2))*(1./(X1-X2))-A*(X1+X2+X4)
      C=(Y4-Y1)/(X4-X1)-A*(X4*X4+X1*X4+X1*X1)-B*(X1+X4)
      D=Y2-X2*(C+X2*(B+A*X2))
      A2=A
      B2=B
      C2=C
      D2=D
      A1=A
      B1=B
      C1=C
      D1=D
C
C     A CUBIC HAS BEEN FIT TO THE CENTER FOUR POINTS AND THAT PART OF
C     THE RESPONSE CURVE IS NOW COMPUTED
C
      L1=IPT(J-1)
      L2=IPT(J)
      DO 10 M=L1,L2
      XM=FLOAT(M)
      RSCV(M)=D+XM*(C+XM*(B+XM*A))
   10 CONTINUE
      IF (IPT(J+1) .EQ. IPT(L)) GO TO 500
  100 X1=FLOAT(IPT(J))
      Y1=PT(J)
      X2=FLOAT(IPT(J+1))
```

```
      Y2=PT(J+1)
      X3=FLOAT(IPT(J+2))
      Y3=PT(J+2)
      S=C+X1*(2.*B+3.*A*X1)
C
C     THIS ADDS A NEW POINT TO THE RIGHT AND COMPUTES THE FITTING SLOPE
C
      A=(((Y3-Y1)/(X3-X1)-S)/(X3-X1))-((Y2-Y1)/(X2-X1)-S)/(X2-X1))/
     1(X3-X2)
      B=((Y3-Y1)/(X3-X1)-S)/(X3-X1)-A*(X3+2.*X1)
      C=(Y3-Y1)/(X3-X1)-A*(X3*X3+ X1*X3+X1*X1)-B*(X3+X1)
      D=Y1-A*X1**3-B*X1*X1-C*X1
C
C     THE NEXT PART OF THE RESPONSE CURVE HAS BEEN COMPUTED
C
      L1=IPT(J)+1
      L2=IPT(J+1)
      DO 20 M=L1,L2
      XM=FLOAT(M)
      RSCV(M)=D+XM*(C+XM*(B+XM*A))
   20 CONTINUE
      A1=A
      B1=B
      C1=C
      D1=D
      J=J+1
      GO TO 10
  500 CONTINUE
      L1=IPT(J)+1
      L2=IPT(J+1)
      DO 400 M=L1,L2
      XM=FLOAT(M)
      RSCV(M)=D+XM*(C+XM*(B+XM*A))
  400 CONTINUE
      IF (NPT .LE. 5) GO TO 600 <-- IF (NPT. EQ. 4) J = J+1
      A=A2
      B=B2
      C=C2
      D=D2
      J=NPT/2+1
  510 CONTINUE
      X1=FLOAT(IPT(J-1))
      Y1=PT(J-1)
      X2=FLOAT(IPT(J-2))
      Y2=PT(J-2)
      X3=FLOAT(IPT(J-3))
      Y3=PT(J-3)
      S=C+X1*(2.*B+3.*A*X1)
C
C     THIS ADDS A POINT TO THE LEFT, COMPUTES THE FITTING SLOPE, AND
C     CALCULATES MORE OF THE RESPONSE CURVE.
C
      A=(((Y3-Y1)/(X3-X1)-S)/(X3-X1))-((Y2-Y1)/(X2-X1)-S)/(X2-X1))/
     1(X3-X2)
      B=((Y3-Y1)/(X3-X1)-S)/(X3-X1)-A*(X3+2.*X1)
      C=(Y3-Y1)/(X3-X1)-A*(X3*X3+X1*X3+X1*X1)-B*(X3+X1)
      D=Y1-A*X1**3-B*X1*X1-C*X1
      L1=IPT(J-2)
      L2=IPT(J-1)-1
      DO 30 M=L1,L2
      XM=FLOAT(M)
```

```
      RSCV(M) =D+XM*(C+XM*(B+XM*A))
   30 CONTINUE
      IF (J .EQ. JK+3) GO TO 600
      J=J-1
      GO TO 510
C
C     NOW EXTRAPOLATE LINEARLY ON BOTH ENDS
C
  600 CONTINUE
      L1=IPT(J-3)
      L2=IPT(J-2)-1
      DO 410 M=L1,L2
      XM=FLOAT(M)
      RSCV(M) =D+XM*(C+XM*(B+XM*A))
  410 CONTINUE
      S=C+FLOAT(IPT(JK))*(2.*B+3.*A*FLOAT(IPT(JK)))
      BS=PT(JK)-S*FLOAT(IPT(JK))
      L2=IPT(JK)-1
      DO 40 M=1,L2
      XM=FLOAT(M)
      RSCV(M) =S*XM+BS
   40 CONTINUE
      S=C1+FLOAT(IPT(L))*(2.*B1+3.*A1*FLOAT(IPT(L)))
      BS=PT(L)-S*FLOAT(IPT(L))
      L1=IPT(L)+1
      DO 50 M=L1,2048
      XM=FLOAT(M)
      RSCV(M) =S*XM+BS
   50 CONTINUE
C
C     TEST TO SEE IF RIGHT OR LEFT ENDS NEED TO BE FLATTENED
C
      RMIN=1.E50
      DO 60 I=600,1448
      IF (RSCV(I) .LT. RMIN) RMIN=RSCV(I)
   60 CONTINUE
      IF (RMIN .LE. 0.) GO TO 70
      DO 70 I=1,2048
      IF (RSCV(I) .GT. 100.*RMIN) RSCV(I) =100.*RMIN
   70 CONTINUE
      RETURN
      END
```