PL-TR-94-2279

SETA/ADS SOFTWARE DEVELOPMENT

Andrew J. Mazzella, Jr.
Kevin P. Larson

RDP Incorporated
391 Totten Pond Road
Waltham, Massachusetts 02154

3 November 1994

Scientific Report No. 6

Approved for public release; distribution unlimited

PHILLIPS LABORATORY
Directorate of Geophysics
AIR FORCE MATERIEL COMMAND
HANSCOM AIR FORCE BASE, MA 01731-3010

"This technical report has been reviewed and is approved for publication"

EDWARD C. ROBINSON
Contract Manager
Data Analysis Division

ROBERT E. McINERNEY, Director
Data Analysis Division

This report has been reviewed by the ESD Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS).

Qualified requestors may obtain additional copies from the Defense Technical Information Center. All others should apply to the National Technical Information Service.

If your address has changed, or if you wish to be removed from the mailing list, or if the addressee is no longer employed by your organization, please notify PL/IM, 29 Randolph Road, Hanscom AFB, MA 01731-3010. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	3 November 1994	Scientific Report No. 6

4. TITLE AND SUBTITLE

SETA/ADS Software Development

5. FUNDING NUMBERS

PE 35160F
PR 7659 TA 05 WU AC

Contract: F19628-89-C-0079

6. AUTHOR(S)

Andrew J. Mazzella, Jr.
Kevin P. Larson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

RDP Incorporated
391 Totten Pond Road
Waltham, Massachusetts 02154

8. PERFORMING ORGANIZATION REPORT NUMBER

RDP-TR-9404

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Phillips Laboratory
29 Randolph Road
Hanscom Air Force Base, Massachusetts 01731-3010
Contract Manager: Edward C. Robinson/GPD

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

PL-TR-94-2279

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Requirements for increased accuracy in the measurement of thermospheric densities and winds prompted the development of the STEP-1 mission, which employed a complement of instruments to measure local densities, compositions and relative velocity components. The triaxial accelerometer (SETA) for this mission could be utilized individually or in conjunction with other instruments to determine local density and wind values, which could then be correlated with geophysical parameters and other geophysical measurements to develop and evaluate thermospheric density and circulation models.

This document describes the data processing flow and software developed to support this mission.

14. SUBJECT TERMS	15. NUMBER OF PAGES
SETA, STEP-1, Density, Thermosphere, Satellite, Winds, Drag, Accelerometer	158
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited

SETA/ADS Software Development

Table of Contents

SETA/ADS Software Development

SETA/ADS Software Development

Introduction

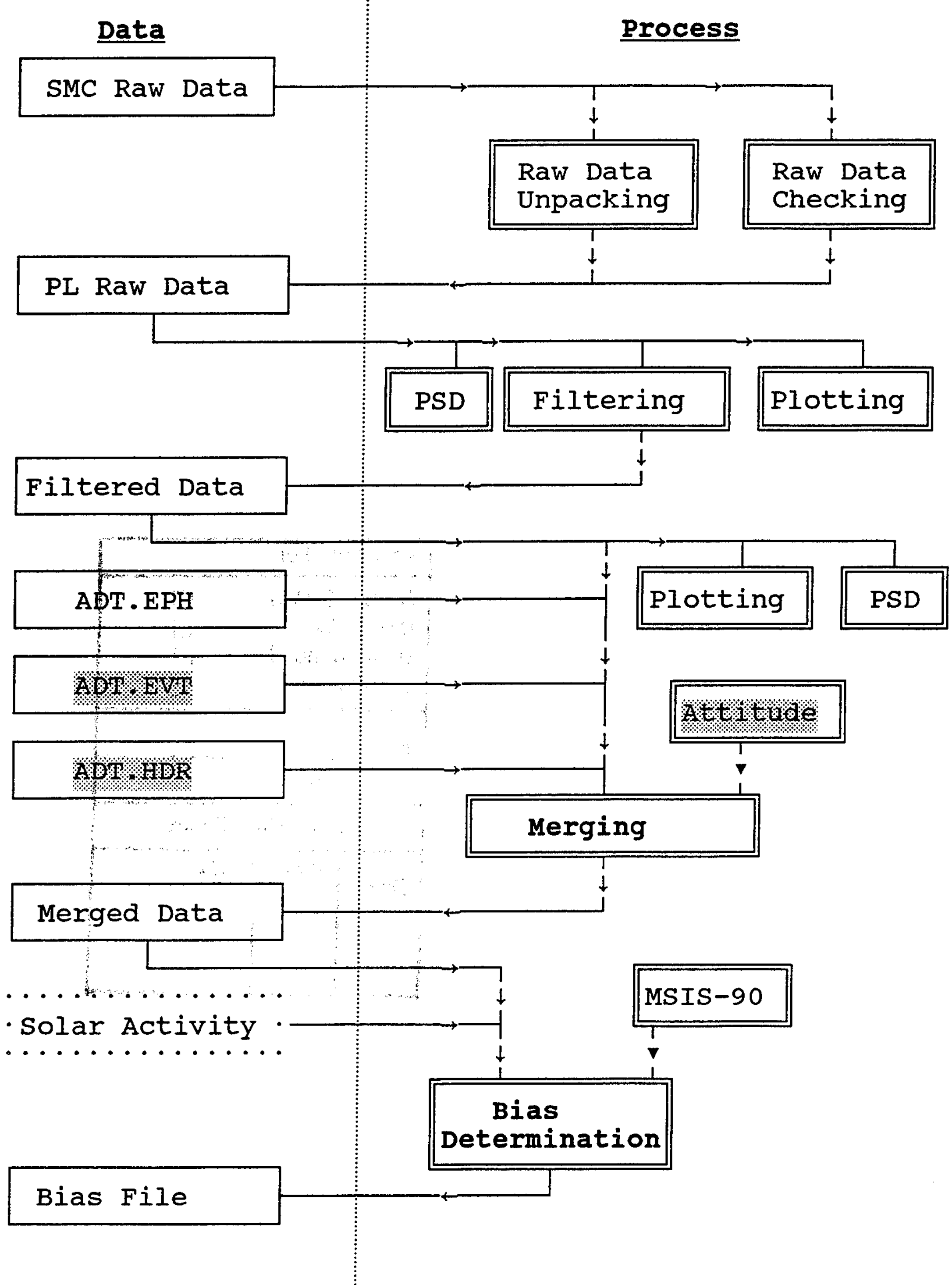
Requirements for increased accuracy in the measurement of thermospheric densities and winds prompted the development of the STEP-1 mission, which employed a complement of instruments to measure local densities, compositions, and relative velocity components. The triaxial accelerometer (SETA) for this mission could be utilized individually or in conjunction with other instruments to determine local density and wind values, which could then be correlated with geographical parameters and other geophysical measurements to develop and evaluate thermospheric density and circulation models.

Due to a second-stage failure, the launch vehicle and STEP-1 payload were destroyed. Because of this, certain portions of the SETA/ADS software system were not finalized. However, this software system could be used as a basis for future missions.

Accesion For	
NTIS CRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

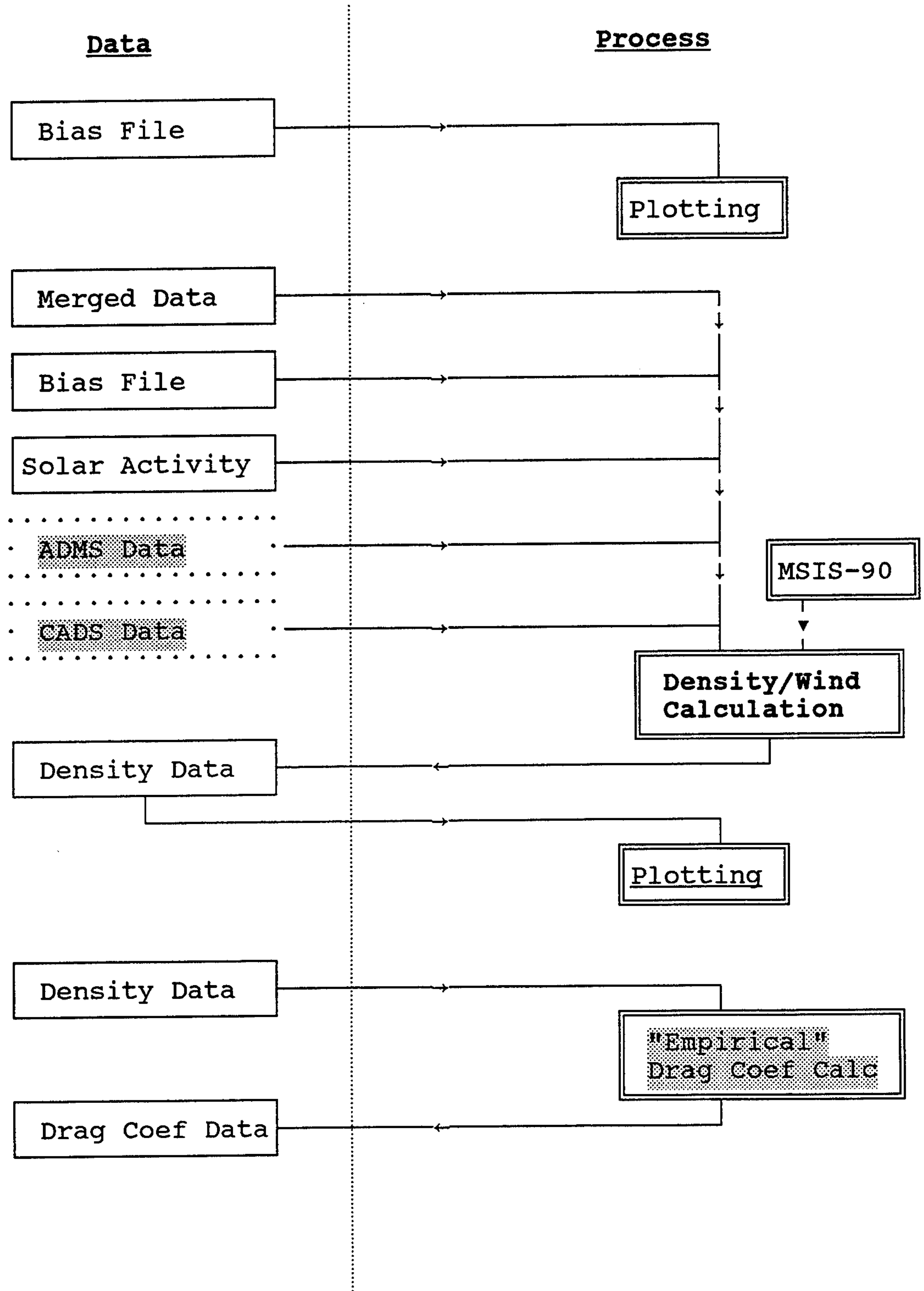
SETA/ADS Software Development

SETA Data Processing Flow



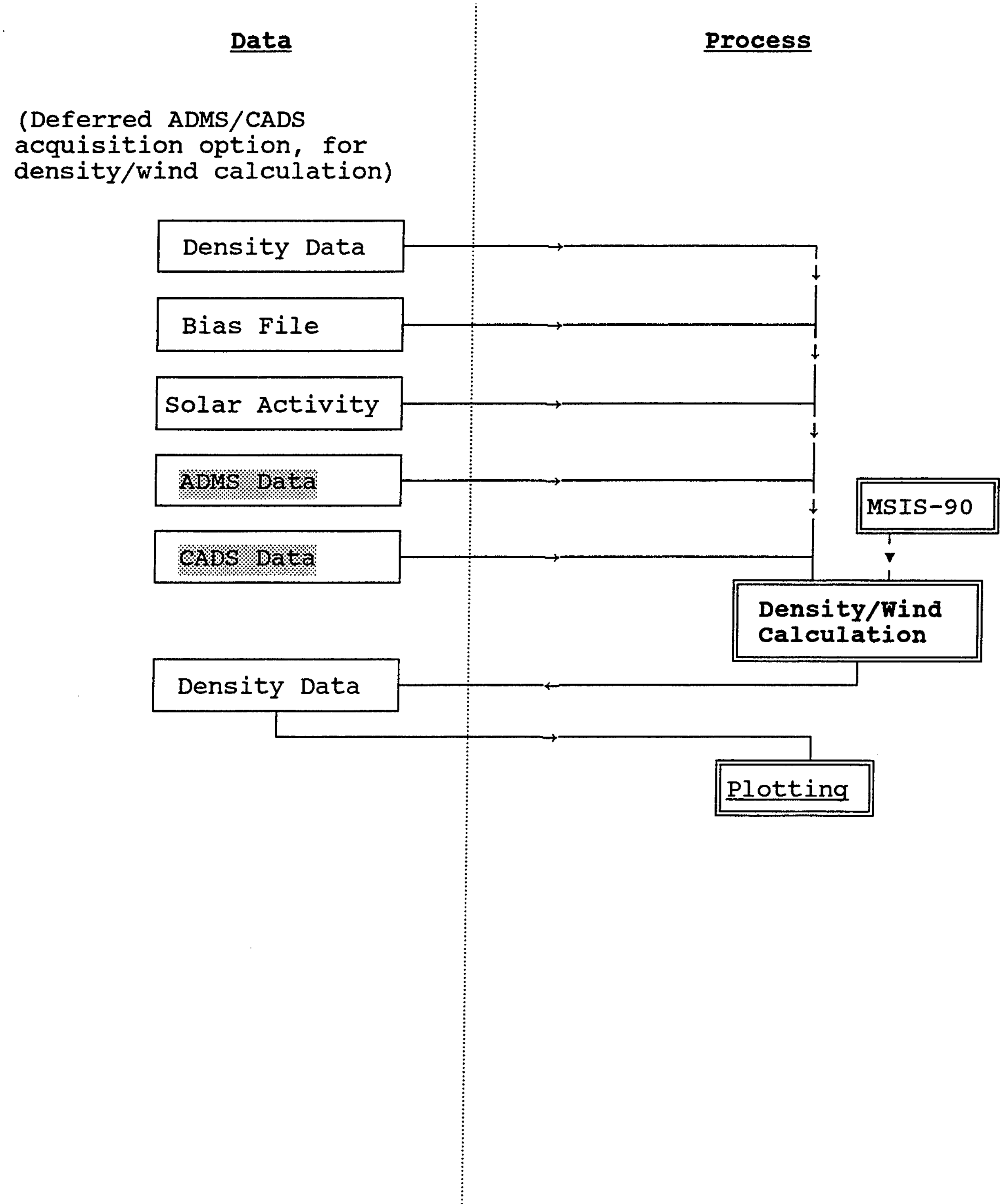
Item - Item developed by PL/GPIM; Item - Item to be defined
Item - Item under development; ···· - Optional input file

SETA/ADS Software Development
SETA Data Processing Flow



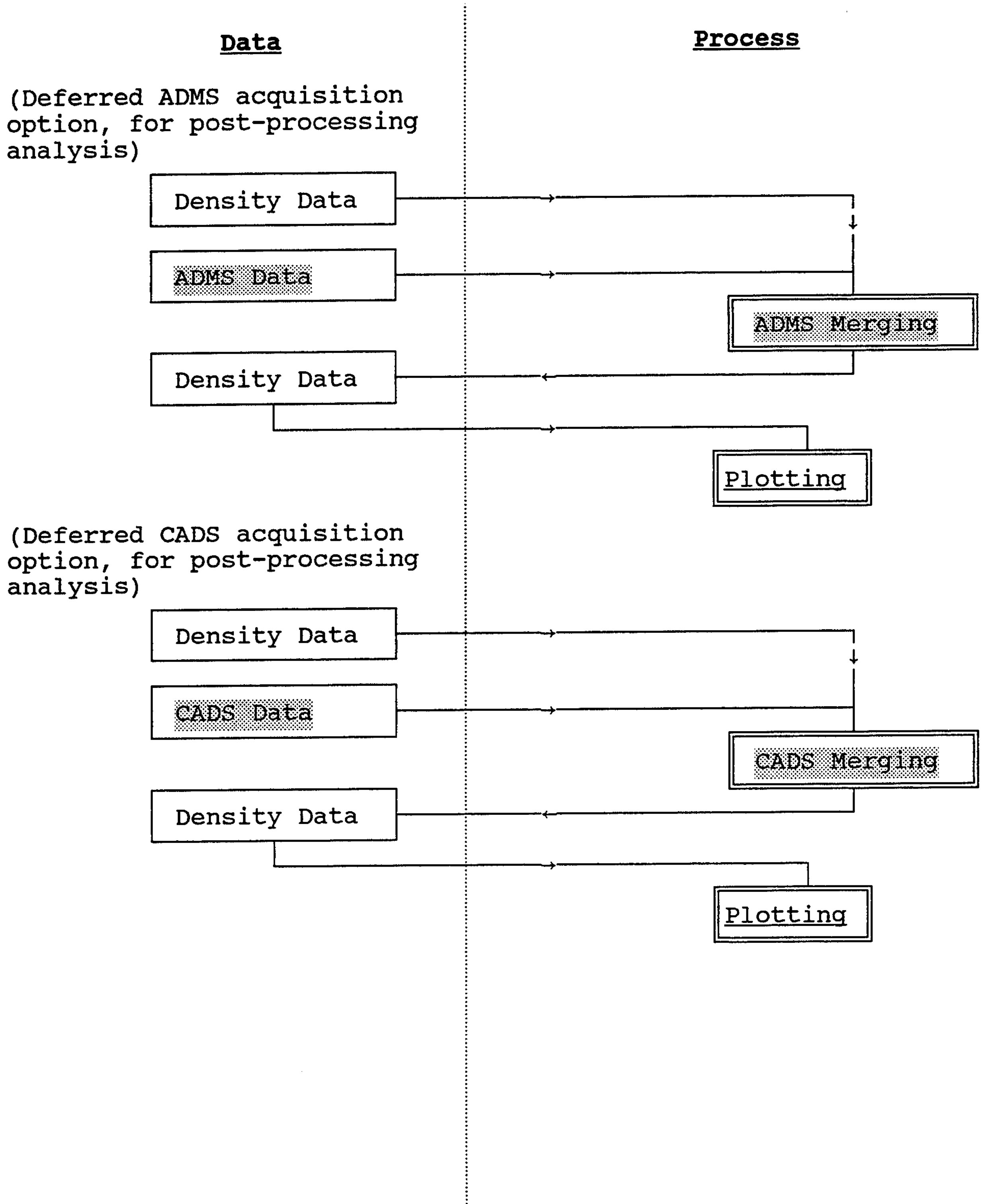
Item - Item developed by PL/GPIM; Item - Item to be defined
Item - Item under development; ···· - Optional input file

SETA/ADS Software Development
SETA Data Processing Flow



Item - Item developed by PL/GPIM; Item - Item to be defined
Item - Item under development; ···· - Optional input file

SETA/ADS Software Development
SETA Data Processing Flow



Item - Item developed by PL/GPIM; Item - Item to be defined
Item - Item under development; ···· - Optional input file

SETA/ADS Software Development

Basic Requirements (dependent on classification of software module):

- Class A: Complete ANSI FORTRAN-77 standard source code, without system-dependent routines or libraries;
- Class B: Complete ANSI FORTRAN-77 standard source code, allowing system-dependent routines or libraries;
- Class C: Basic ANSI FORTRAN-77 standard source code, allowing extensions compatible with Fortran-90 and system-dependent routines or libraries;
- Class D: All system-specific features allowed.

Standard program header format:

- PROGRAM/SUBROUTINE/FUNCTION statement;
- Comments for description of routine, with source file name, creation date, and author's name;
- Edit history, earliest first, with revised file name, edit date, and editor's name;
- Data and specification file descriptions, as appropriate.

Unless otherwise indicated, file formats will be FORTRAN 'UNFORMATTED', with the detailed binary structure being dependent on the system-specific implementations. PL data files are planned to be generated spanning approximately a single day of data, coinciding with the SMC SETA data interval.

Raw Data Unpacking Program

Class B - designated for VAX/VMS and 486/DOS.

Acquire data in SMC format and convert to scaled integer acceleration values, blocked in groups of up to 600 values for each accelerometer axis (one minute of data), with a range flag and sensor temperature for each sample and a day/time tag for each block, adjusting for the on-board delay (approximately 0.15 seconds).

An ASCII file header for the generated data file will contain an experiment identifier, the calendar date (year, month, day) of the start of the data segment, with the associated SETA day number, the SETA day number for generation of the file, and a numeric factor (ASCALE) for converting the scaled integer acceleration values to engineering units (micro-Gees). The numeric conversion factor will have a default value of 0.01 (micro-Gees per integer unit), but this value will an adjustable input value for the program.

The input SMC data format is described on page 19, and the generated PL raw data format is described on page 20. The Raw Data Unpacking Program listing is provided in Appendix A.

Notes:

1. The SETA day number is the number of days since December 31, 1989, so that January 1, 1990 is SETA day number 1.
2. Values of ASCALE appropriate for the three accelerometer operating ranges are:

Range	ASCALE
"A"	1.00
"B"	0.10
"C"	0.01

Raw Data Checking Program

Class B - designated for VAX/VMS and 486/DOS.
Acquire data in SMC format and report beginning and ending times for continuous data segments, at the 10 Hz sampling rate. Report accelerometer range changes, acceleration saturation (counts = +2047 or counts = -2048), and temperatures outside the nominal operating range ([0°C, 50°C]). By user option, list 2-minute averages and standard deviations for accelerations and temperature versus time.

Also generate an unpacked accelerometer data file, in the same format generated by the Raw Data Unpacking program, with the same provision for assigning an acceleration scale factor.

The input SMC data format is described on page 19, and the generated PL raw data format is described on page 20. The Raw Data Checking Program listing is provided in Appendix B.

Enhancements:

Allow for optional on-screen plot displays for acceleration and temperature values versus time, by integrating the Raw Data Plot routines. Time ranges and data decimation should be user-specifiable parameters.

Acceleration PSD Program

Class D - designated for VAX/VMS, with secondary application for 486/DOS.

Acquire a designated number of accelerometer data samples (restricted to be an integral power of two) in PL Raw Data format at a specified starting time, convert these to acceleration units (micro-Gees), and perform a Power Spectral Density (PSD) analysis using the classic periodogram techniques. Plot the PSD for each accelerometer axis requested, using user-specified or default ranges for the frequencies and amplitudes.

Allow provisions for accepting the PL Filter Data format as the accelerometer data source, in which case the plot caption should also display the filtering parameters with the plot for each axis.

The nominal maximum for the number of acceleration samples to acquire for a PSD is 4096 (2^{12}), but this value should be defined as a parameter for possible revision. As with the Filtering program, provisions should be incorporated for editing of wild points and interpolating across time gaps in the data, with options to enable or disable these features. Such editing should be reported to the user when it is performed for a designated data segment.

Enhancements:

Allow provisions for generating a sequence of PSD's for a designated initial time, incremented by the sample size for each member of the sequence. In this mode, allow the option to perform an average of the PSD amplitudes over the sequence, and to plot the average PSD.

The input PL raw data format is described on page 20, and the input PL filtered data format is described on page 21. The Acceleration PSD Program listing is provided in Appendix C.

Filtering Program

Class C - designated for VAX/VMS.
Acquire acceleration and temperature data in the PL Raw Data format and perform time-centered digital filtering of the data, according to the parameters specified by the user and the methods specified in the processing algorithm. Use DC extension at the beginning and end of the data segment to initialize and terminate the filtering, accounting for the possible occurrence of "wild point" values at these limits.

Incorporate editing provisions for "wild points" and data gaps, as enabled by the user, with user specification of the thresholds for "wild points" and allowed time gaps.

Separate filtering techniques may be implemented for the temperature data, in contrast to the acceleration data.

The input PL raw data format is described on page 20, and the generated PL filtered data format is described on page 21. The Filtering Program listing is provided in Appendix D.

Density and Winds Check-out Program

Class C - designated for 486/DOS, but with some designated Class A modules, for later incorporation into production Density/Wind program.

Acquire data in either PL Raw Data format or PL Filtered Data format, and, using a Keplerian model orbit and a linearized drag coefficient model, calculate density and wind estimates based on the measured accelerations and estimated biases. Generate summary listings of the resulting density and winds, and store these values in a file with the same form as the PL Raw/Filtered Data format, with the density estimate replacing the Temperature and the wind components replacing the corresponding Accelerometer values. An estimated altitude, in tenths of kilometers, will replace the Range Flag values.

Parameters required for the density and wind estimates will be:

- Semi-major axis of orbit (km);
- Orbital eccentricity;
- Orbital inclination (degrees);
- Latitude of perigee (degrees);
- Time since perigee (seconds);
- Satellite mass (kg);
- SETA-X bias estimate (micro-Gees);
- SETA-Y bias estimate (micro-Gees);
- SETA-Z bias estimate (micro-Gees);
- Reference frontal area for satellite (square meters);
- Zero-order in-track drag coefficient;
- First-order in-track drag coefficient;
- First-order cross-track drag coefficient;
- Reference velocity for drag coefficient (km/sec).

A standard shape will be assumed for the earth, for altitude calculations, and a nominal attitude will be assumed for the satellite.

The input PL raw data format is described on page 20, the PL filtered data format is described on page 21, and the generated PL density check-out format is described on page 23.

Ephemeris and Attitude Merge Program

Class C - designated for VAX/VMS.

Acquire data in the PL Filtered Data format and determine the associated attitude and ephemeris values, as acquired from the SMC/TRW Attitude routine and Ephemeris data. The ephemeris data may need to be interpolated to match the sample times for the accelerometer data, according to algorithms provided by SMC or developed for this application **(TBD)**. Discrete quantities, stored as flags, will be matched by the nearest sample in time.

Incorporate provisions to decimate the accelerometer data sequence by a specified integer factor, prior to calculating the associated attitude and ephemeris parameters. (This factor would be chosen by the user, consistent with the filtering parameters used for the data.)

As part of the processing report, list the day, time, orbit number, altitude, and orbital leg (based on the sign of the radial component of the velocity) for the beginning and end (pairwise) of each continuous segment of data within the processing sequence, according to a time gap criterion specified by the user. Generate a separate list of the day, time, orbit number, altitude, and orbital leg for thruster firings, indicating which thrusters are active. These reports will assist in evaluating bias determinations and density/wind measurements.

The input PL filtered data format is described on page 21, the input Agency ephemeris format is described on page 24, the input Agency event format is described on page 26, the input Agency header format is described on page 28, and the generated PL merged data format is described on page 29. The Ephemeris and Attitude Merge Program listing is provided in Appendix E.

Notes:

1. The software interfaces for the ephemeris and attitude routines will be defined for compatibility with the ADMS data processing as well as for the SETA processing.

Bias Determination Program

Class C - designated for VAX/VMS.

For each orbit, determine the appropriate time interval for evaluating the accelerometer biases. The time interval will be determined by the following criteria, in order of preference:

1) A period of 500 continuous seconds centered on apogee, provided that this period is more than 10 minutes after the accelerometer has been activated;
2) A period of 500 continuous seconds containing apogee, provided that this period is more than 10 minutes after the accelerometer has been activated;
3) Pairwise downleg/upleg intervals (in order of preference):
   a) A period of 500 continuous downleg (upleg) seconds, provided that this period is more than 10 minutes after the accelerometer has been activated, and the measurements occur above an altitude of 300 kilometers;
   b) A composite period of 500 downleg (upleg) seconds, provided that this period begins more than 10 minutes after the accelerometer has been activated, and the measurements occur above an altitude of 300 kilometers;
   c) A composite period of at least 60 downleg (upleg) seconds, but as large as possible (up to 500 seconds) within the constraints of at least 10 minutes after activation of the accelerometer and above an altitude of 300 kilometers.

If none of the above conditions can be satisfied, then no biases will be calculated for that particular orbit. Otherwise, the biases will be calculated based on:

1. the filtered drag acceleration, for altitudes above 500 km, assuming no drag;
2. the difference between the filtered drag accelerations and those predicted by the MSIS-90 model, for altitudes below 500 km.

The results will be appended to the SETA bias file, together with the median altitude, the mean accelerometer temperature, and the number of data samples used.

Mean molecular weights and ambient temperatures as reported by the MSIS-90 model will be used for the calculation of the drag coefficients. The satellite surface temperatures, for the thermal accommodation calculation, will be obtained from **(TBD)**.

As part of the report for the bias calculation, list the average accelerometer values used for the calculation and, if utilized, the drag values computed from the MSIS-90 model, as well as the parameters reported to the bias file.

The input PL merged data format is described on page 29, the input PL solar activity data format (for the MSIS-90 calculations) is described on page 33, and the generated PL bias data format is described on page 34. The Bias Determination Program listing is provided in Appendix F.

Notes:

1. For initial development, the accelerometer will be assumed to be operational full-time (although not always in data collection mode). For part-time operation, the activation time of the accelerometer will be determined from the command indicator in the SMC Event file.
2. Typical altitude ranges for 500-second intervals during various orbital segments are:

Orbital Segment	Altitude Variation
Apogee-centered (1500 km)	15 km
Apogee-bounded	95 km
Data Initiation (915 km)	350 km
Drag Detection (500 km-centered)	315 km
Bias Cutoff (300 km)	225 km

Bias Plotting Program

Class C - designated for VAX/VMS.

On a set of five stacked plots with coordinated time axes, plot the bias values for each accelerometer axis as individual points, together with the mean altitude for the bias calculation and the mean accelerometer operating temperature (in degrees Celsius). Allow for automatic or user-defined ranges for the time, bias, altitude, and temperature values. Allow provisions for the incorporation of a routine representing a fit or functional representation of the bias values, to be plotted as a continuous line.

The input bias data format is described on page 34.

Density and Winds Calculation Program

Class C - designated for VAX/VMS.

Based on the measured accelerations, bias values, and vehicle dynamical effects (rotation and angular accelerations), calculate the corresponding neutral thermospheric density and relative velocity components. Adjust the relative velocity components to account for the known vehicle and diurnal rotation effects to determine thermospheric winds. The wind components will be calculated in instantaneous vehicle coordinates and stored in accelerometer coordinates (to accommodate the error analysis), with software provisions to transform these values to Geographic polar coordinates, based on the satellite location and attitude.

Bias values will be calculated for each sample time based on interpolations or fits **(TBD)** from the SETA Bias Data file. The MSIS-90 model will be used to provide mean molecular weights and ambient temperatures for the drag coefficient calculation, and these model values will be stored with the drag coefficients and the model density, for each accelerometer sample. Provisions will be incorporated to utilize densities and mean molecular weights from the ADMS, or densities, mean molecular weights, gas temperatures, and in-track wind speeds from the CADS, instead of model or accelerometer values, if specified by the user. The satellite surface temperatures, for the thermal accommodation calculation, will be obtained from **(TBD)**.

Because of the likelihood of ADMS or CADS data acquisition subsequent to density and wind processing, the Density and Winds Calculation program will allow for use of either the Merged Data format or the Density Data format as input.

The input PL merged data format is described on page 29, the input PL solar activity data format (for the MSIS-90 calculations) is described on page 33, the reference bias data format is described on page 34, the reference PL ADMS data format is described on page 35, the reference CADS data format is described on page 36, and the generated PL density data format is described on page 37. The Density and Winds Calculation Program listing is provided in Appendix G.

ADMS Merge Program

Class C - designated for VAX/VMS.

Acquire data from the processed ADMS format **(TBD)** and store densities and mean molecular weights from the ADMS into the PL density data format, superseding any existing ADMS data. If the PL density file header indicates that the ADMS data had been used for density or wind calculations, then clear the header processing date for the density and wind calculation, and also clear (zero) the associated accelerometer density and wind results.

The input ADMS data format is described on page 35, and the PL density data format is described on page 37.

CADS Merge Program

Class C - designated for VAX/VMS.

Acquire data from the processed CADS format **(TBD)** and store densities, mean molecular weights, gas temperatures, and in-track wind speeds from the CADS into the PL density data format, superseding any existing CADS data. If the PL density file header indicates that the CADS data had been used for density or wind calculations, then clear the header processing date for the density and wind calculation, and also clear (zero) the associated accelerometer density and wind results.

The input CADS data format is described on page 36, and the PL density data format is described on page 37.

SMC SETA Format

Data
(For a stream of time tagged groups)

Item	Description	Type
1.	Time Tag Year (since 1900)	I*1
2.	Time Tag Month	I*1
3.	Time Tag Day (of Month)	I*1
4.	Time Tag Hour	I*1
5.	Time Tag Minute	I*1
6.	Time Tag Second	I*1
For I = 1 to 10 (0 msec to 900 msec after Time Tag)		
7.	Packed SETA Acceleration and Temperature Sample	C*7

Notes:
1. Integer fields are stored with the LS byte in the lower address and the MS byte in the higher address.
2. The SETA acceleration and temperature packing is as follows:

X-Acceleration Field		Y-Acceleration Field		Z-Acceleration Field		Temperature Field
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6

with each 14-bit field decomposed as follows:

Data Value											Sign	Range	
10	9	8	7	6	5	4	3	2	1	0	0	1	0

The second row indicates the power of two for each bit weight. A non-zero Sign bit indicates that 2048 should be subtracted from the Data Value. The Range value maps to a range designation according to:
0: "A" range (least sensitive)
1: "B" range
2: "C" range (most sensitive)
3: unused
The Temperature Range value is fixed as 2.

SETA/ADS Software Development

PL SETA Raw Data Format

Header

Item	Description	Type
1. EXPID	Experiment Identifier ('SETA-5')	C*8
2. DTYPE	Data Type ('RAW ')	C*8
3. SAMPRT	Nominal Data Sampling Rate (Hz) (≡ 10)	C*8
4. DECIM	Decimation Factor for Data Segment (≡ 1)	C*4
5. ASCALE	Scale Factor for Acceleration Data (to micro-Gees)	C*8
6. BEGYR	Year Number for Beginning of Data Segment	C*4
7. BEGMON	Month Number for Beginning of Data Segment	C*2
8. BEGDAY	Day of Month Number for Beginning of Data Segment	C*2
9. BEGDN	SETA Day Number for Beginning of Data Segment	C*4
10. GENDN	SETA Day Number for Processing of Raw Data Segment	C*4
11. BLANK(11+8*K)	Blank Words (to match Filter Header)	C*4
Total		96+32*K=224 bytes

Data

Item	Description	Type
1. DATDN	SETA Day Number for Beginning of Data Block	I*2
2. DATTIM	Time of Day for Beginning of Data Block, in tenths of seconds	I*4
3. NSAMP	Number of Time Samples in Data Block (≤ 600)	I*4
For I = 1 to NSAMP		
4. ACCX(I)	SETA-X (STEP -Y) Acceleration Scaled Units	I*2
5. ACCY(I)	SETA-Y (STEP -Z) Acceleration Scaled Units	I*2
6. ACCZ(I)	SETA-Z (STEP +X) Acceleration Scaled Units	I*2
7. TEMP(I)	SETA Temperature, in Degrees	I*2
8. RNGFLG(I)	Accelerometer Range Flags (Packed)	I*2
Total		20 to 6010 bytes

Variable Types are:
- C = character
- I = integer
- R = floating point
- Z = complex
- *n = number of bytes

Notes:

1. The range flags are embedded in half-bytes (nibbles) of the RNGFLG word, with the X-range flag as the lowest order nibble, the Y-range and Z-range flags as successively higher order nibbles, and the highest order nibble as zero.

MS Nibble			LS Nibble
0	Z-range	Y-range	X-range

2. The numeric conventions for the range flags are the same as those for the SMC data ("A" = 0; "B" = 1; "C" = 2).
3. The number of filter parameters (K) is 4.
4. NSAMP is nominally equal to 600, but can be smaller for blocks just prior to a time gap or the end of the data segment.

SETA/ADS Software Development

PL SETA Filtered Data Format

Header

Item	Description	Type
1. EXPID	Experiment Identifier ('SETA-5')	C*8
2. DTYPE	Data Type ('FILTER ')	C*8
3. SAMPRT	Nominal Data Sampling Rate (Hz) (≡ 10)	C*8
4. DECIM	Decimation Factor for Data Segment	C*4
5. ASCALE	Scale Factor for Acceleration Data (to micro-Gees)	C*8
6. BEGYR	Year Number for Beginning of Data Segment	C*4
7. BEGMON	Month Number for Beginning of Data Segment	C*2
8. BEGDAY	Day of Month Number for Beginning of Data Segment	C*2
9. BEGDN	SETA Day Number for Beginning of Data Segment	C*4
10. GENDN	SETA Day Number for Processing of Raw Data Segment	C*4
11. FILTDN	SETA Day Number for Filtering of Data Segment	C*4
12. TGAP	Time Gap Allowance (Seconds) for Interpolating	C*4
13. WPTHRX	Wild Point Threshold: X-Acceleration (micro-Gees)	C*8
14. WPTHRY	Wild Point Threshold: Y-Acceleration (micro-Gees)	C*8
15. WPTHRZ	Wild Point Threshold: Z-Acceleration (micro-Gees)	C*8
16. WPTHRT	Wild Point Threshold: Temperature (°C)	C*8
17. WPEDX	Wild Point Editing Flag: X-Acceleration	C*1
18. WPEDY	Wild Point Editing Flag: Y-Acceleration	C*1
19. WPEDZ	Wild Point Editing Flag: Z-Acceleration	C*1
20. WPEDT	Wild Point Editing Flag: Temperature	C*1
21. FILTX(K)	X-Acceleration Filter Parameters	C*8
22. FILTY(K)	Y-Acceleration Filter Parameters	C*8
23. FILTZ(K)	Z-Acceleration Filter Parameters	C*8
24. FILTT(K)	Temperature Filter Parameters	C*8
Total		96+32*K=224 bytes

Data

Item	Description	Type
1. DATDN	SETA Day Number for Beginning of Data Block	I*2
2. DATTIM	Time of Day for Beginning of Data Block, in tenths of seconds	I*4
3. NSAMP	Number of Time Samples in Data Block (≤ 600)	I*4
For I = 1 to NSAMP		
4. ACCX(I)	SETA-X (STEP -Y) Filtered Acceleration Scaled Units	I*2
5. ACCY(I)	SETA-Y (STEP -Z) Filtered Acceleration Scaled Units	I*2
6. ACCZ(I)	SETA-Z (STEP +X) Filtered Acceleration Scaled Units	I*2
7. TEMP(I)	SETA Filtered Temperature, in Degrees	I*2
8. RNGFLG(I)	Accelerometer Range Flags (Packed)	I*2
Total		20 to 6010 bytes

Notes:

1. The range flags are embedded in half-bytes (nibbles) of the RNGFLG word, with the X-range flag as the lowest order nibble and the Y-range and Z-range flags as successively higher order nibbles. The highest order nibble has the appropriate bit set to one if the data value results from interpolation (from a wild point or sampling gap).

MS Nibble			LS Nibble
T ¦ Z ¦ Y ¦ X	Z-range	Y-range	X-range

2. The numeric conventions for the range flags are the same as those for the SMC data ("A" = 0; "B" = 1; "C" = 2).

3. The filtering for the accelerometer axes will have the same form but possibly different parametric values for each axis.
4. The filter for the temperature values can be different in form from the filters for the accelerometer values. (A median filter is a likely form.)
5. The number of filter parameters (K) is 4.
6. The wild point editing flags are set to 'Y' if editing has been enabled and are set to 'N' if editing has been disabled.
7. NSAMP is nominally equal to 600, but can be smaller for blocks just prior to a time gap or the end of the data segment.
8. The time interval between data samples in the Data Block is DECIM/SAMPRT, in seconds.

SETA/ADS Software Development

PL SETA Density Check-out Format

Header

Item	Description	Type
1. EXPID	Experiment Identifier ('SETA-5')	C*8
2. DTYPE	Data Type ('CHECKOUT')	C*8
3. SAMPRT	Nominal Data Sampling Rate (Hz) (≡ 10)	C*8
4. DECIM	Decimation Factor for Data Segment (≡ 1)	C*4
5. ASCALE	Scale Factor for Acceleration Data (to micro-Gees)	C*8
6. BEGYR	Year Number for Beginning of Data Segment	C*4
7. BEGMON	Month Number for Beginning of Data Segment	C*2
8. BEGDAY	Day of Month Number for Beginning of Data Segment	C*2
9. BEGDN	SETA Day Number for Beginning of Data Segment	C*4
10. GENDN	SETA Day Number for Processing of Raw Data Segment	C*4
11. FILTDN	SETA Day Number for Filtering of Data Segment	C*4
12. TGAP	Time Gap Allowance (Seconds) for Interpolating	C*4
13. WPTHRX	Wild Point Threshold: X-Acceleration (micro-Gees)	C*8
14. WPTHRY	Wild Point Threshold: Y-Acceleration (micro-Gees)	C*8
15. WPTHRZ	Wild Point Threshold: Z-Acceleration (micro-Gees)	C*8
16. WPTHRT	Wild Point Threshold: Temperature (°C)	C*8
17. WPEDX	Wild Point Editing Flag: X-Acceleration	C*1
18. WPEDY	Wild Point Editing Flag: Y-Acceleration	C*1
19. WPEDZ	Wild Point Editing Flag: Z-Acceleration	C*1
20. WPEDT	Wild Point Editing Flag: Temperature	C*1
21. FILTX(K)	X-Acceleration Filter Parameters	C*8
22. FILTY(K)	Y-Acceleration Filter Parameters	C*8
23. FILTZ(K)	Z-Acceleration Filter Parameters	C*8
24. FILTT(K)	Temperature Filter Parameters	C*8
Total		96+32*K=224 bytes

Data

Item	Description	Type
1. DATDN	SETA Day Number for Beginning of Data Block	I*2
2. DATTIM	Time of Day for Beginning of Data Block, in tenths of seconds	I*4
3. NSAMP	Number of Time Samples in Data Block (≤ 600)	I*4
For I = 1 to NSAMP		
4. EWINDX(I)	SETA-X (STEP -Y) Wind Estimate, in m/sec	I*2
5. EWINDY(I)	SETA-Y (STEP -Z) Wind Estimate, in m/sec	I*2
6. EWINDZ(I)	SETA-Z (STEP +X) Wind Estimate, in m/sec	I*2
7. DENS(I)	SETA Density Estimate (10^{-15} g/cm^3)	I*2
8. SALT(I)	Estimated Satellite Altitude, in tenths of km	I*2
Total		20 to 6010 bytes

Variable Types are:

C = character
I = integer
R = floating point
Z = complex
*n = number of bytes

Notes:

1. The header parameters associated with the filtering processing will be left as blanks if the Raw Data format is used as the input data source.
2. The number of filter parameters (K) is 4.
3. NSAMP is nominally equal to 600, but can be smaller for blocks just prior to a time gap or the end of the data segment.

SETA/ADS Software Development

Agency Data Tape Ephemeris Format
(from TRW Software Design document)

Data
(For a stream of time tagged groups, reported once per minute)

Item	Description	Type
1. IYY	Time Tag Year (since 1900)	I*1
2. IMM	Time Tag Month	I*1
3. IDD	Time Tag Day (of Month)	I*1
4. IHH	Time Tag Hour	I*1
5. IMN	Time Tag Minute	I*1
6. ISS	Time Tag Second	I*1
7. MJDAY	Modified Julian Day (standard Julian Day - 2400000.5)	I*4
8. UTMSEC	Universal Time in milliseconds	I*4
9. XECI	Satellite X-position in meters (ECI, mean equinox of date)	I*4
10. YECI	Satellite Y-position in meters (ECI, mean equinox of date)	I*4
11. ZECI	Satellite Z-position in meters (ECI, mean equinox of date)	I*4
12. VXECI	Satellite X-velocity in millimeters/second (ECI, mean equinox of date)	I*4
13. VYECI	Satellite Y-velocity in millimeters/second (ECI, mean equinox of date)	I*4
14. VZECI	Satellite Z-velocity in millimeters/second (ECI, mean equinox of date)	I*4
15. RMAG	Radius vector magnitude to satellite (from earth center) in meters	I*4
16. ALT	Satellite altitude in meters, from reference ellipsoid	I*4
17. GLAT	Geocentric latitude in micro-degrees	I*4
18. GLON	Geocentric longitude in micro-degrees, positive East	I*4
19. VMAG	Velocity vector magnitude in millimeters/second	I*4
20. LT	Local time in hours times 10^6	I*4
21. GMR	Satellite radial position in earth-centered dipole geomagnetic coordinates (in 10^3×EMR)	I*4
22. GMLAT	Satellite latitude in earth-centered dipole geomagnetic coordinates (micro-degrees)	I*4
23. GMLON	Satellite longitude in earth-centered dipole geomagnetic coordinates (micro-degrees, positive East from meridian containing South geographic pole)	I*4
24. SMR	Satellite radial position in earth eccentric dipole solar magnetic coordinates (in 10^3×EMR)	I*4
25. SMLAT	Satellite latitude in earth eccentric dipole solar magnetic coordinates (micro-degrees)	I*4
26. SMLT	Satellite local time in earth eccentric dipole solar magnetic coordinates (hours times 10^6)	I*4
27. GSMR	Satellite radial position in earth eccentric dipole solar magnetospheric coordinates (in 10^3×EMR)	I*4
28. GSMLAT	Satellite latitude in earth eccentric dipole solar magnetospheric coordinates (micro-degrees)	I*4
29. GSMLT	Satellite local time in earth eccentric dipole solar magnetospheric coordinates (hours times 10^6)	I*4
30. BMAG	Magnitude of model magnetic field in milli-gammas	I*4
31. BXECI	Model magnetic field ECI X-component in pico-Tesla	I*4
32. BYECI	Model magnetic field ECI Y-component in pico-Tesla	I*4
33. BZECI	Model magnetic field ECI Z-component in pico-Tesla	I*4
34. GMLT	Geomagnetic local time in hours times 10^6	I*4
35. SOLANG	Geocentric angle between satellite and sun in micro-degrees	I*4
36. INVLAT	L-shell invariant latitude parameter in micro-degrees	I*4
37. BFILATN	Geocentric latitude in micro-degrees for 100 km northern field line trace intercept	I*4
38. BFILONN	Geocentric longitude (+E) in micro-degrees for 100 km northern field line trace intercept	I*4
39. BFILATS	Geocentric latitude in micro-degrees for 100 km southern field line trace intercept	I*4
40. BFILONS	Geocentric longitude (+E) in micro-degrees for 100 km southern field line trace intercept	I*4
41. LSHELL	L-shell parameter in 10^6 times EMR	I*4
42. BMIN	Minimum field strength along current magnetic field line in pico-Tesla	I*4
43. BMLAT	Geocentric latitude for minimum magnetic field strength location along current field line (micro-degrees)	I*4

SETA/ADS Software Development

Agency Data Tape Ephemeris Format (continued)

Data

Item	Description	Type
44. BMLON	Geocentric longitude for minimum magnetic field strength location along current field line (micro-degrees)	I*4
45. BMRAD	Geocentric radial coordinate for minimum magnetic field strength location along current field line (meters)	I*4
46. BCNJLAT	Conjugate point geocentric latitude in micro-degrees	I*4
47. BCNJLON	Conjugate point geocentric longitude in micro-degrees	I*4
48. BCNJRAD	Conjugate point geocentric radial coordinate in meters	I*4
49. SOLECIX	Solar X-coordinate in kilometers (ECI)	I*4
50. SOLECIY	Solar Y-coordinate in kilometers (ECI)	I*4
51. SOLECIZ	Solar Z-coordinate in kilometers (ECI)	I*4
52. LUNECIX	Lunar X-coordinate in kilometers (ECI)	I*4
53. LUNECIY	Lunar Y-coordinate in kilometers (ECI)	I*4
54. LUNECIZ	Lunar Z-coordinate in kilometers (ECI)	I*4
55. GRA	Right ascension of Greenwich mean sidereal time in micro-degrees	I*4
56. BFIMAGN	Magnetic field magnitude in pico-Tesla for 100 km northern field line trace intercept	I*4
57. BFIMAGS	Magnetic field magnitude in pico-Tesla for 100 km southern field line trace intercept	I*4
58. BMECIX	Dipole field moment X-component in pico-Tesla (ECI)	I*4
59. BMECIY	Dipole field moment Y-component in pico-Tesla (ECI)	I*4
60. BMECIZ	Dipole field moment Z-component in pico-Tesla (ECI)	I*4
61. BOECIX	Eccentric dipole offset X-component in meters (ECI)	I*4
62. BOECIY	Eccentric dipole offset Y-component in meters (ECI)	I*4
63. BOECIZ	Eccentric dipole offset Z-component in meters (ECI)	I*4

Notes:

1. The reference date for the Modified Julian day is November 16, 1858.
2. The reference ellipsoid for the earth is given by

$$R_e = A_e \times \sqrt{\frac{1 + \tan^2\phi}{1 + \frac{\tan^2\phi}{(1-f)^2}}}$$

   where
   R_e = reference ellipsoid radius (km),
   A_e = equatorial earth radius (6378.135 km),
   f = ellipsoidal flattening (1/298.26),
   ϕ = geocentric latitude.
3. EMR is a unit of distance in terms of the Earth Mean Radius (6371.2 km).
4. The invariant latitude ψ is given by

$$\cos^2(\psi) = \frac{1}{L}$$

   for the L-shell parameter 'L' in mean earth radii.

SETA/ADS Software Development

Agency Data Tape Event File Format

Header

Item	Description	Type
1. IVYY	Time Tag Year (since 1900)	I*1
2. IVMM	Time Tag Month	I*1
3. IVDD	Time Tag Day (of month)	I*1
4. IVHH	Time Tag Hour	I*1
5. IVMN	Time Tag Minute	I*1
6. IVSS	Time Tag Second	I*1
7. IVYYSC	Scheduled Time Tag Year (since 1900)	I*1
8. IVMMSC	Scheduled Time Tag Month	I*1
9. IVDDSC	Scheduled Time Tag Day (of month)	I*1
10. IVHHSC	Scheduled Time Tag Hour	I*1
11. IVMNSC	Scheduled Time Tag Minute	I*1
12. IVSSSC	Scheduled Time Tag Second	I*1
13. IDCMD	Command ID	I*1
14. IDPROC	Processor ID	I*1
15. PARAM(16)	Parameters	I*1

HSC Data Structure

Item	Description	Type
1. ERCODE	Error Code	I*1 (?)
2. TYPE(4)	Data Transfer Type	I*4
3. EXPHIST	Experiment Sat History	C*7

ACS Data Structure

Item	Description	Type
1. TACH.TDIF	Tachometer Data: Time Delta	I*1
2. TACH.COUNT(3)	Tachometer Data: Tachometer Count	I*2
3. WHLM.TDIF	Wheel Momentum Data: Time Delta	I*1
4. WHLM.CMD(2)	Wheel Momentum Data: Wheel Momentum Command	I*4
5. WHLC.TDIF	Wheel Control Data: Time Delta	I*1
6. WHLC.CNTRL(3)	Wheel Control Data: Wheel Control	I*2
7. ALGTAM(2).TDIF	Algorithm TAM (Magnetometer) Data: Time Delta	I*1
8. ...COUNT(3)	Algorithm TAM (Magnetometer) Data: TAM Count	I*2
9. TORQ(2).TDIF	Torqrod Data: Time Delta	I*1
10. ...ACT(3)	Torqrod Data: Torqrod On	I*2
11. SCNERR.TDIF	Scan Error Data: Time Delta	I*1
12. SCNERR.ROLL	Scan Error Data: Roll	I*4
13. SCNERR.PITCH	Scan Error Data: Pitch	I*4
14. SCNPUL.TDIF	Scan Pulse: Time Delta	I*1
15. SCNPUL.NPH	Scan Pulse: Normal Phase	I*4
16. SCNPUL.WIDTH(4,2)	Scan Pulse: Width	I*2
17. ECI.TDIF	ECI Data: Time Delta	I*1
18. ECI.POS(3)	ECI Data: ECI Position	I*4
19. EPHEM.ALT	Ephemeris: Altitude	I*4
20. EPHEM.FPANG	Ephemeris: Flight Path Angle	I*4
21. EPHEM.EPHERR	Ephemeris: Ephemeris Error	I*1
22. ATTEST(3).TDIF	Attitude Estimation Data: Time Delta	I*1
23. ...RATE(2)	Attitude Estimation Data: Rate	I*4
24. ...ERR(2)	Attitude Estimation Data: Error	I*4
25. TAM(15).TDIF	TAM: Time Delta	I*1
26. ...COUNT(3)	TAM: Count	I*2
27. MOMENT.TDIF	Momentum: Time Delta	I*1
28. MOMENT.INHIB	Momentum: Inhibit Flag	I*1
29. MOMENT.ERREST(3)	Momentum: Error Estimate	I*4
30. ATTERR.TDIF	Attitude Error: Time Delta	I*1
31. ATTERR.INT(2)	Attitude Error: Integral	I*4
32. ACS.TDIF	ACS Thruster Data: Time Delta	I*1
33. ACS.THRUST(4)	ACS Thruster Data: ACS Thruster PW	I*2

ACS Data Structure (continued)

Item	Description	Type
34. DELV(3).TDIF	Delta V Data: Time Delta	I*1
35. ...TLEFT	Delta V Data: Delta V Time Left	I*2
36. STATUS.TDIF	Status: Time Delta	I*1
37. STATUS.BADREV(2)	Status: Bad Rev Count	I*2
38. STATUS.EARTH(2)	Status: Earth Presence Count	I*1
39. STATUS.OLDSUM	Status: Old Sum Count	I*1
40. STATUS.OLDES	Status: Old ES Data	I*1
41. STATUS.MODE	Status: Mode	I*1
42. ATTIT.TDIF	Attitude: Time Delta	I*1
43. ATTIT.ESRERR	Attitude: Earth Sensor Roll Error	I*4
44. ATTIT.ESPERR	Attitude: Earth Sensor Pitch Error	I*4
45. ATTIT.ESRRATE	Attitude: Earth Sensor Roll Error Rate	I*4
46. ATTIT.ESPRATE	Attitude: Earth Sensor Pitch Error Rate	I*4
47. FILL	Fill	C*44

CDH Data Structure

Item	Description	Type
1. ERCODE	Error Code	I*2
2. PORT	Port 6000	I*1

Notes:

1. Processor IDs in Header are:
   - 2: HSC
   - 25: ACS
   - 26: CDH
2. The notation

   X(n).A
   ...B

   denotes an iterated structure, with the sequence {X.A, X.B} repeated "n" times, in contrast to an ordinary sequence

   X(n).A
   X(n).B

   in which "n" successive occurrences of X.A are followed by "n" successive occurrences of X.B.

SETA/ADS Software Development

Agency Data Tape Header File Format

Header

Item	Description	Type
1. IRON	Mission IRON	C*4
2. BTSTART	Universal Time for Start of Data (YYMMDDHHMMSS Format)	6xI*1
3. BTSTOP	Universal Time for Stop of Data (YYMMDDHHMMSS Format)	6xI*1
4. RVSTART	Starting Revolution (at Ascending Node)	I*4
5. RVSTOP	Stopping Revolution (at Ascending Node)	I*4
6. CRDATE	File Creation Date (YYMMDD) {DD-MON-YYYY ?}	C*11
7. ADTFCOM	ADTF Operations Comments	C*800
8. ERRSUM	Summary of Errors	C*5600
9. NCONT	Number of Telemetry Downlinks Used to Collect Data	I*4
10. NREV	Number of Revolutions in File	I*4

Data

Item	Description	Type
For I = 1 to NCONT		
1. BTCONT(I)	Universal Time of Start of Contact (YYMMDDHHMMSS Format)	6xI*1
2. STATION(I)	Station for Contact Data	C*7
3. PKPROC(I)	Number of 64 KB Packets Received During Contact	I*4
4. PKERR(I)	Number of 64 KB Packets Received in Error	I*4
5. COMM1(I)	Comments	C*80
6. COMM2(I)	Comments	C*80
For J = 1 to NREV		
7. KREV(J)	Revolution Number	I*4
8. BTASCN(J)	Time of Passage of Ascending Node (YYMMDDHHMMSS Format)	6xI*1
9. BTPERI(J)	Perigee Time (YYMMDDHHMMSS Format)	6xI*1
For K = 1 to 2		
10. ECLOBS(J,K)	Percentage of Sun Eclipsed by Earth	I*4
11. PENBEG(J,K)	Penumbra Start Time (YYMMDDHHMMSS Format)	6xI*1
12. UMBBEG(J,K)	Umbra Start Time (YYMMDDHHMMSS Format)	6xI*1
13. UMBEND(J,K)	Umbra End Time (YYMMDDHHMMSS Format)	6xI*1
14. PENEND(J,K)	Penumbra End Time (YYMMDDHHMMSS Format)	6xI*1

SETA/ADS Software Development

PL SETA Merged Data Format

Header

Item	Description	Type
1. EXPID	Experiment Identifier ('SETA-5')	C*8
2. DTYPE	Data Type ('MERGE ')	C*8
3. SAMPRT	Nominal Data Sampling Rate (Hz) (≡ 10)	C*8
4. DECIM	Decimation Factor for Data Segment	C*4
5. ASCALE	Scale Factor for Acceleration Data (to micro-Gees)	C*8
6. BEGYR	Year Number for Beginning of Data Segment	C*4
7. BEGMON	Month Number for Beginning of Data Segment	C*2
8. BEGDAY	Day of Month Number for Beginning of Data Segment	C*2
9. BEGDN	SETA Day Number for Beginning of Data Segment	C*4
10. GENDN	SETA Day Number for Processing of Raw Data Segment	C*4
11. FILTDN	SETA Day Number for Filtering of Data Segment	C*4
12. MRGDN	SETA Day Number for Merging of Data Segment	C*4
13. DENDN	SETA Day Number for Density/Wind Processing (Blank)	C*4
14. TGAP	Time Gap Allowance (Seconds) for Interpolating	C*4
15. WPTHRX	Wild Point Threshold: X-Acceleration (micro-Gees)	C*8
16. WPTHRY	Wild Point Threshold: Y-Acceleration (micro-Gees)	C*8
17. WPTHRZ	Wild Point Threshold: Z-Acceleration (micro-Gees)	C*8
18. WPTHRT	Wild Point Threshold: Temperature (^{o}C)	C*8
19. WPEDX	Wild Point Editing Flag: X-Acceleration	C*1
20. WPEDY	Wild Point Editing Flag: Y-Acceleration	C*1
21. WPEDZ	Wild Point Editing Flag: Z-Acceleration	C*1
22. WPEDT	Wild Point Editing Flag: Temperature	C*1
23. FILTX(K)	X-Acceleration Filter Parameters	C*8
24. FILTY(K)	Y-Acceleration Filter Parameters	C*8
25. FILTZ(K)	Z-Acceleration Filter Parameters	C*8
26. FILTT(K)	Temperature Filter Parameters	C*8
27. AREF	Reference Area for Drag Coefficient (m^2)	C*8
28. POS1	Accelerometer Location: STEP-X Coordinate (mm)	C*8
29. POS2	Accelerometer Location: STEP-Y Coordinate (mm)	C*8
30. POS3	Accelerometer Location: STEP-Z Coordinate (mm)	C*8
31. CALOPT	Calculation Option for Densities and Winds (Blank)	C*2
Total		138+32*K=266 bytes

Data

Item	Description	Type
1. NSAMP	Number of Time Samples in Data Block (≤ 64)	I*4
For I = 1 to NSAMP		
2. DATDN(I)	SETA Day Number for Data Sample	I*2
3. DATTIM(I)	Time of Day for Data Sample, in tenths of seconds	I*4
4. ACCX(I)	SETA-X (STEP -Y) Filtered Acceleration Scaled Units	I*2
5. ACCY(I)	SETA-Y (STEP -Z) Filtered Acceleration Scaled Units	I*2
6. ACCZ(I)	SETA-Z (STEP +X) Filtered Acceleration Scaled Units	I*2
7. TEMP(I)	SETA Filtered Temperature, in Degrees	I*2
8. RNGFLG(I)	Accelerometer Range Flags (Packed)	I*2
9. ORBNUM(I)	Orbit Number	I*2
10. ALT(I)	Vehicle Altitude (m)	I*4
11. LAT(I)	Vehicle Latitude (hundredths of a degree)	I*2
12. LON(I)	Vehicle Longitude (hundredths of a degree)	I*2
13. RAD(I)	Local Orbit Radius (m)	I*4
14. GMLAT(I)	Geomagnetic Latitude (hundredths of a degree)	I*2
15. GMLON(I)	Geomagnetic Longitude (hundredths of a degree)	I*2
16. GMLT(I)	Geomagnetic Local Time (tenths of seconds)	I*2
17. VRAD(I)	Vehicle Radial Velocity (Inertial Coordinates, m/sec)	I*2
18. VTHETA(I)	Vehicle Latitudinal Velocity, Positive South (Inertial Coordinates, m/sec)	I*2
19. VPHI(I)	Vehicle Longitudinal Velocity, Positive East (Inertial Coordinates, m/sec)	I*2
20. SOLRA(I)	Solar Right Ascension (degrees)	I*2
21. SOLDEC(I)	Solar Declination (degrees)	I*2
22. ECLSTA(I)	Eclipse Status	I*1
23. ECLPCT(I)	Peak Percentage Eclipse	I*1

SETA/ADS Software Development

PL SETA Merged Data Format (continued)

Data

Item	Description	Type
24. ATTP(I)	Pitch Attitude Angle (arc-min)	I*2
25. ATTY(I)	Yaw Attitude Angle (arc-min)	I*2
26. ATTR(I)	Roll Attitude Angle (arc-min)	I*2
27. ROTP(I)	Pitch Attitude Rate (arc-sec/sec)	I*2
28. ROTY(I)	Yaw Attitude Rate (arc-sec/sec)	I*2
29. ROTR(I)	Roll Attitude Rate (arc-sec/sec)	I*2
30. SMASS(I)	Vehicle Mass (kg)	I*2
31. CG1(I)	Vehicle Center-of-Mass STEP-X Coordinate (mm)	I*2
32. CG2(I)	Vehicle Center-of-Mass STEP-Y Coordinate (mm)	I*2
33. CG3(I)	Vehicle Center-of-Mass STEP-Z Coordinate (mm)	I*2
34. I11(I)	Vehicle Moment of Inertia: I_{xx} (kg-cm^2) (STEP coordinates)	I*4
35. I22(I)	Vehicle Moment of Inertia: I_{yy} (kg-cm^2) (STEP coordinates)	I*4
36. I33(I)	Vehicle Moment of Inertia: I_{zz} (kg-cm^2) (STEP coordinates)	I*4
37. I12(I)	Vehicle Moment of Inertia: I_{xy} (kg-cm^2) (STEP coordinates)	I*4
38. I13(I)	Vehicle Moment of Inertia: I_{xz} (kg-cm^2) (STEP coordinates)	I*4
39. I23(I)	Vehicle Moment of Inertia: I_{yz} (kg-cm^2) (STEP coordinates)	I*4
40. THRFLG(I)	Thruster Firing Flags (Packed)	I*2
41. TRQFLG(I)	Torqrod Excitation Flags (Packed)	I*2
Total		100 to 6148 bytes

Notes:
1. The Merged Data Header format is the same as that for the Density Data Header, with only differences in the header word contents, including the use of blank words.
2. The range flags are embedded in half-bytes (nibbles) of the RNGFLG word, with the X-range flag as the lowest order nibble and the Y-range and Z-range flags as successively higher order nibbles. The highest order nibble has the appropriate bit set to one if the data value results from interpolation (from a wild point or sampling gap).

MS Nibble		LS Nibble
T : Z : Y : X	Z-range	Y-range	X-range

3. The numeric conventions for the range flags are the same as those for the SMC data ("A" = 0; "B" = 1; "C" = 2).
4. The filtering for the accelerometer axes will have the same form but possibly different parametric values for each axis.
5. The filter for the temperature values can be different in form from the filters for the accelerometer values. (A median filter is a likely form.)
6. The number of filter parameters (K) is 4.
7. The wild point editing flags are set to 'Y' if editing has been enabled and are set to 'N' if editing has been disabled.
8. NSAMP is nominally equal to 64, but can be smaller for blocks just prior to a time gap or the end of the data segment.
9. The time interval between data samples in the Data Block is DECIM/SAMPRT, in seconds.
10. The range for longitude, geomagnetic longitude, solar Right

PL SETA Merged Data Format
(continued)

Notes (continued):

Ascension, and roll angle values will be -180° to +180° (rather than 0° to 360°).

11. The thruster firing flags are embedded in nibbles of the THRFLG word, with the thrusters numbered 1 = +pitch, 2 = -pitch, 3 = -yaw, and 4 = +yaw, and the flag value set to 1 (or 2 or 3?) when the thruster is firing.

MS Nibble			LS Nibble
Thruster 4	Thruster 3	Thruster 2	Thruster 1

12. The Torqrod excitation flags are embedded in nibbles of the TRQFLG word, with the Torqrods numbered **(TBD)**, and the flag value set to 1 (+20 $A\text{-}m^2$ dipole) or 2 (-20 $A\text{-}m^2$ dipole) when the Torqrod is excited.

MS Nibble			LS Nibble
0	Torqrod 3	Torqrod 2	Torqrod 1

13. If only approximate values for the center of mass or moments of inertia are available, or these quantities change only sufficiently slowly, they may be stored in the header block rather than the individual data blocks.
14. The Eclipse Status will be designated by:
    0: no current eclipse
    1: penumbral eclipse phase
    2: umbral eclipse phase

Source Items for Merged Data

Merge Data Item	Source Item(s)	Notes
MERGE.2:DATDN	FILTER.2:DATDN	
MERGE.3:DATTIM	FILTER.3:DATTIM	Adjusted for decimation
MERGE.4:ACCX	FILTER.4:ACCX	Nearest sample (for decimation)
MERGE.5:ACCY	FILTER.5:ACCY	Nearest sample (for decimation)
MERGE.6:ACCZ	FILTER.6:ACCZ	Nearest sample (for decimation)
MERGE.7:TEMP	FILTER.7:TEMP	Nearest sample (for decimation)
MERGE.8:RNGFLG	FILTER.8:RNGFLG	Nearest sample (for decimation)
MERGE.9:ORBNUM	HDR.7:KREV, HDR.8:BTASCN	Nearest previous sample
MERGE.10:ALT	EPHEM.16:ALT	Interpolate to data sample time
MERGE.11:LAT	EPHEM.17:GLAT	Interpolate to data sample time
MERGE.12:LON	EPHEM.18:GLON	Interpolate to data sample time
MERGE.13:RAD	EPHEM.15:RMAG	Interpolate to data sample time
MERGE.14:GMLAT	EPHEM.22:GMLAT	Interpolate to data sample time
MERGE.15:GMLON	EPHEM.23:GMLON	Interpolate to data sample time
MERGE.16:GMLT	EPHEM.34:GMLT	Interpolate to data sample time
MERGE.17:VRAD	EPHEM.12:VXECI,EPHEM.13:VYECI,EPHEM.14:VZECI	Interpolate to data sample time and transform
MERGE.18:VTHETA	EPHEM.9:XECI,EPHEM.10:YECI,EPHEM.11:ZECI,EPHEM.12:VXECI,EPHEM.13:VYECI,EPHEM.14:VZECI	Interpolate to data sample time and transform

SETA/ADS Software Development

PL SETA Merged Data Format
(continued)

Source Items for Merged Data
(continued)

Merge Data Item	Source Item(s)	Notes
MERGE.19:VPHI	EPHEM.9:XECI,EPHEM.10:YECI,EPHEM.11:ZECI,EPHEM.12:VXECI,EPHEM.13:VYECI,EPHEM.14:VZECI	Interpolate to data sample time and transform
MERGE.20:SOLRA	EPHEM.49:SOLECIX, EPHEM.50:SOLECIY, EPHEM.51:SOLECIZ	Interpolate to data sample time and transform
MERGE.21:SOLDEC	EPHEM.49:SOLECIX, EPHEM.50:SOLECIY, EPHEM.51:SOLECIZ	Interpolate to data sample time and transform
MERGE.22:ECLSTA	HDR.11:PENBEG, HDR.12:UMBBEG, HDR.13:UMBEND, HDR.14:PENEND	Nearest sample
MERGE.23:ECLPCT	HDR.10:ECLOBS	Nearest sample
MERGE.24:ATTP		Interpolate to data sample time
MERGE.25:ATTY		Interpolate to data sample time
MERGE.26:ATTR		Interpolate to data sample time
MERGE.27:ROTP		Interpolate to data sample time
MERGE.28:ROTY		Interpolate to data sample time
MERGE.29:ROTR		Interpolate to data sample time
MERGE.30:SMASS		Interpolate to data sample time
MERGE.31:CG1		Interpolate to data sample time
MERGE.32:CG2		Interpolate to data sample time
MERGE.33:CG3		Interpolate to data sample time
MERGE.34:I11		Interpolate to data sample time
MERGE.35:I22		Interpolate to data sample time
MERGE.36:I33		Interpolate to data sample time
MERGE.37:I12		Interpolate to data sample time
MERGE.38:I13		Interpolate to data sample time
MERGE.39:I23		Interpolate to data sample time
MERGE.40:THRFLG	EVTACS.32	Nearest sample
MERGE.41:TRQFLG	EVTACS.10	Nearest sample

SETA/ADS Software Development

PL Solar Activity Data Format

The Solar Activity Data file will be an ASCII file with each record (line) formatted as follows:

Item	Description	Format
1. MYR	Four-digit year for date of data	I5
2. MON	Two-digit month number for date of data	I3
3. MDAY	Two-digit day-of-month for date of data	I3
4. APDAILY	Daily Geomagnetic Activity Parameter A_p	I4
For I = 1 to 8		
5. AP(I)	Individual 3-hr Geomagnetic Activity Parameter A_p Values for the Day	8*I4
6. FLUX	Daily Solar Flux Parameter $F_{10.7}$	F6.1
7. FLUXAV	Solar Flux Parameter $F_{10.7}$ Centered Average for 90 Days	F6.1
Total		59 characters

Notes:

1. The individual 3-hour A_p values are for the periods commencing at 0, 3, 6, 9, 12, 15, 18, and 21 hours (UT).
2. The solar flux unit is 10^{-22} w/(m^2-Hz-sec).

SETA/ADS Software Development

PL Bias Data Format

The Bias Data file will be an ASCII file with each record (line) formatted as follows:

Item	Description	Format
1. ORBNUM	Orbit Number, at median time	I6
2. BIASDN	SETA Day Number for Bias Value	I5
3. BIASTM	Time of Day for Median Time of Bias Samples (seconds)	I6
4. NSAMP	Number of Data Samples Used for Bias Calculation	I6
5. ALT	Altitude for Median Time of Bias Samples (km)	F7.1
6. BIASX	SETA X-bias (micro-Gees)	F9.3
7. BIASY	SETA Y-bias (micro-Gees)	F9.3
8. BIASZ	SETA Z-bias (micro-Gees)	F9.3
9. TEMP	Accelerometer Temperature (^{o}C)	F6.1
Total		63 characters

SETA/ADS Software Development

ADMS Data Format

Data

Item	Description	Type

(To Be Specified)

SETA/ADS Software Development

CADS Data Format

Data

Item	Description	Type

(To Be Specified)

SETA/ADS Software Development

PL SETA Density Data Format

Header

Item	Description	Type
1. EXPID	Experiment Identifier ('SETA-5')	C*8
2. DTYPE	Data Type ('DENSITY ')	C*8
3. SAMPRT	Nominal Data Sampling Rate (Hz) (≡ 10)	C*8
4. DECIM	Decimation Factor for Data Segment	C*4
5. ASCALE	Scale Factor for Acceleration Data (to micro-Gees)	C*8
6. BEGYR	Year Number for Beginning of Data Segment	C*4
7. BEGMON	Month Number for Beginning of Data Segment	C*2
8. BEGDAY	Day of Month Number for Beginning of Data Segment	C*2
9. BEGDN	SETA Day Number for Beginning of Data Segment	C*4
10. GENDN	SETA Day Number for Processing of Raw Data Segment	C*4
11. FILTDN	SETA Day Number for Filtering of Data Segment	C*4
12. MRGDN	SETA Day Number for Merging of Data Segment	C*4
13. DENDN	SETA Day Number for Density/Wind Processing	C*4
14. TGAP	Time Gap Allowance (Seconds) for Interpolating	C*4
15. WPTHRX	Wild Point Threshold: X-Acceleration (micro-Gees)	C*8
16. WPTHRY	Wild Point Threshold: Y-Acceleration (micro-Gees)	C*8
17. WPTHRZ	Wild Point Threshold: Z-Acceleration (micro-Gees)	C*8
18. WPTHRT	Wild Point Threshold: Temperature (°C)	C*8
19. WPEDX	Wild Point Editing Flag: X-Acceleration	C*1
20. WPEDY	Wild Point Editing Flag: Y-Acceleration	C*1
21. WPEDZ	Wild Point Editing Flag: Z-Acceleration	C*1
22. WPEDT	Wild Point Editing Flag: Temperature	C*1
23. FILTX(K)	X-Acceleration Filter Parameters	C*8
24. FILTY(K)	Y-Acceleration Filter Parameters	C*8
25. FILTZ(K)	Z-Acceleration Filter Parameters	C*8
26. FILTT(K)	Temperature Filter Parameters	C*8
27. AREF	Reference Area for Drag Coefficient (m^2)	C*8
28. POS1	Accelerometer Location: STEP-X Coordinate (mm)	C*8
29. POS2	Accelerometer Location: STEP-Y Coordinate (mm)	C*8
30. POS3	Accelerometer Location: STEP-Z Coordinate (mm)	C*8
31. CALOPT	Calculation Option for Densities and Winds	C*2
Total		138+32*K=266 bytes

Data

Item	Description	Type
1. NSAMP	Number of Time Samples in Data Block (≤ 48)	I*4
For I = 1 to NSAMP		
2. DATDN(I)	SETA Day Number for Data Sample	I*2
3. DATTIM(I)	Time of Day for Data Sample, in tenths of seconds	I*4
4. ACCX(I)	SETA-X (STEP -Y) Filtered Acceleration Scaled Units	I*2
5. ACCY(I)	SETA-Y (STEP -Z) Filtered Acceleration Scaled Units	I*2
6. ACCZ(I)	SETA-Z (STEP +X) Filtered Acceleration Scaled Units	I*2
7. TEMP(I)	SETA Filtered Temperature, in Degrees	I*2
8. RNGFLG(I)	Accelerometer Range Flags (Packed)	I*2
9. ORBNUM(I)	Orbit Number	I*2
10. ALT(I)	Vehicle Altitude (m)	I*4
11. LAT(I)	Vehicle Latitude (hundredths of a degree)	I*2
12. LON(I)	Vehicle Longitude (hundredths of a degree)	I*2
13. RAD(I)	Local Orbit Radius (m)	I*4
14. GMLAT(I)	Geomagnetic Latitude (hundredths of a degree)	I*2
15. GMLON(I)	Geomagnetic Longitude (hundredths of a degree)	I*2
16. GMLT(I)	Geomagnetic Local Time (tenths of seconds)	I*2
17. VRAD(I)	Vehicle Radial Velocity (Inertial Coordinates, m/sec)	I*2
18. VTHETA(I)	Vehicle Latitudinal Velocity, Positive South (Inertial Coordinates, m/sec)	I*2
19. VPHI(I)	Vehicle Longitudinal Velocity, Positive East (Inertial Coordinates, m/sec)	I*2
20. SOLRA(I)	Solar Right Ascension (degrees)	I*2
21. SOLDEC(I)	Solar Declination (degrees)	I*2
22. ECLSTA(I)	Eclipse Status	I*1
23. ECLPCT(I)	Peak Percentage Eclipse	I*1

PL SETA Density Data Format (continued)

Data

Item	Description	Type
24. ATTP(I)	Pitch Attitude Angle (arc-min)	I*2
25. ATTY(I)	Yaw Attitude Angle (arc-min)	I*2
26. ATTR(I)	Roll Attitude Angle (arc-min)	I*2
27. ROTP(I)	Pitch Attitude Rate (arc-sec/sec)	I*2
28. ROTY(I)	Yaw Attitude Rate (arc-sec/sec)	I*2
29. ROTR(I)	Roll Attitude Rate (arc-sec/sec)	I*2
30. SMASS(I)	Vehicle Mass (kg)	I*2
31. CG1(I)	Vehicle Center-of-Mass STEP-X Coordinate (mm)	I*2
32. CG2(I)	Vehicle Center-of-Mass STEP-Y Coordinate (mm)	I*2
33. CG3(I)	Vehicle Center-of-Mass STEP-Z Coordinate (mm)	I*2
34. I11(I)	Vehicle Moment of Inertia: I_{xx} (kg-cm^2) (STEP coordinates)	I*4
35. I22(I)	Vehicle Moment of Inertia: I_{yy} (kg-cm^2) (STEP coordinates)	I*4
36. I33(I)	Vehicle Moment of Inertia: I_{zz} (kg-cm^2) (STEP coordinates)	I*4
37. I12(I)	Vehicle Moment of Inertia: I_{xy} (kg-cm^2) (STEP coordinates)	I*4
38. I13(I)	Vehicle Moment of Inertia: I_{xz} (kg-cm^2) (STEP coordinates)	I*4
39. I23(I)	Vehicle Moment of Inertia: I_{yz} (kg-cm^2) (STEP coordinates)	I*4
40. THRFLG(I)	Thruster Firing Flags (Packed)	I*2
41. TRQFLG(I)	Torqrod Excitation Flags (Packed)	I*2
42. APDAILY(I)	Daily Geomagnetic Activity Parameter A_p × 10	I*2
43. APCURR(I)	Current 3-hr Geomagnetic Activity Parameter A_p × 10	I*2
44. AP3HP(I)	Three Hours Prior 3-hr Geomagnetic Activity Parameter A_p × 10	I*2
45. AP6HP(I)	Six Hours Prior 3-hr Geomagnetic Activity Parameter A_p × 10	I*2
46. AP9HP(I)	Nine Hours Prior 3-hr Geomagnetic Activity Parameter A_p × 10	I*2
47. AP1DPAV(I)	24 Hours Prior (Centered) Average Geomagnetic Activity Parameter A_p × 10	I*2
48. AP2DPAV(I)	48 Hours Prior (Centered) Geomagnetic Activity Parameter A_p × 10	I*2
49. FLUXPR(I)	Solar Flux Parameter $F_{10.7}$ for Previous Day × 10	I*2
50. FLUXAV(I)	Solar Flux Parameter $F_{10.7}$ Centered Average for 90 Days × 10	I*2
51. CD1(I)	STEP-X Drag Coefficient × 1000	I*2
52. CD2(I)	STEP-Y Drag Coefficient × 1000	I*2
53. CD3(I)	STEP-Z Drag Coefficient × 1000	I*2
54. BIASX(I)	SETA X-bias (Scaled Acceleration Units)	I*2
55. BIASY(I)	SETA Y-bias (Scaled Acceleration Units)	I*2
56. BIASZ(I)	SETA Z-bias (Scaled Acceleration Units)	I*2
57. MDEN(I)	MSIS-90 Density (10^{-15} g/cm^3)	I*2
58. MWT(I)	MSIS-90 Mean Molecular Weight in AMU × 1000	I*2
59. MTEMP(I)	MSIS-90 Ambient Temperature (°K)	I*2
60. ADEN(I)	ADMS Measured Density (10^{-15} g/cm^3)	I*2
61. AWT(I)	ADMS Measured Mean Molecular Weight in AMU × 1000	I*2
62. CDEN(I)	CADS Measured Density (10^{-15} g/cm^3)	I*2
63. CWT(I)	CADS Measured Mean Molecular Weight in AMU × 1000	I*2
64. CTEMP(I)	CADS Measured Gas Temperature (°K)	I*2
65. CWIND(I)	CADS Measured In-track Wind (m/sec)	I*2
66. DENO(I)	Measured Zero-order Density from Accelerometer (10^{-15} g/cm^3)	I*2
67. DEN(I)	Measured Density from Accelerometer (10^{-15} g/cm^3)	I*2
68. WINDX(I)	Measured X-Wind (SETA Coordinates, m/sec)	I*2
69. WINDY(I)	Measured Y-Wind (SETA Coordinates, m/sec)	I*2
70. WINDZ(I)	Measured Z-Wind (SETA Coordinates, m/sec)	I*2
Total		158 to 7396 bytes

Notes:

1. The range flags are embedded in half-bytes (nibbles) of the RNGFLG word, with the X-range flag as the lowest order nibble and the Y-range and Z-range flags as successively higher order nibbles. The highest order nibble has the appropriate bit set to one if the data value results from interpolation

Notes (continued):

(from a wild point or sampling gap).

MS Nibble			LS Nibble
T ¦ Z ¦ Y ¦ X	Z-range	Y-range	X-range

2. The numeric conventions for the range flags are the same as those for the SMC data ("A" = 0; "B" = 1; "C" = 2).
3. The filtering for the accelerometer axes will have the same form but possibly different parametric values for each axis.
4. The filter for the temperature values can be different in form from the filters for the accelerometer values. (A median filter is a likely form.)
5. The number of filter parameters (K) is 4.
6. The wild point editing flags are set to 'Y' if editing has been enabled and are set to 'N' if editing has been disabled.
7. NSAMP is nominally equal to 48, but can be smaller for blocks just prior to a time gap or the end of the data segment.
8. The time interval between data samples in the Data Block is DECIM/SAMPRT, in seconds.
9. The range for longitude, geomagnetic longitude, solar Right Ascension, and roll angle values will be -180° to +180° (rather than 0° to 360°).
10. The thruster firing flags are embedded in nibbles of the THRFLG word, with the thrusters numbered 1 = +pitch, 2 = -pitch, 3 = -yaw, and 4 = +yaw, and the flag value set to 1 (or 2 or 3?) when the thruster is firing.

MS Nibble			LS Nibble
Thruster 4	Thruster 3	Thruster 2	Thruster 1

11. The Torqrod excitation flags are embedded in nibbles of the TRQFLG word, with the Torqrods numbered **(TBD)**, and the flag value set to 1 (+20 A-m^2 dipole) or 2 (-20 A-m^2 dipole) when the Torqrod is excited.

MS Nibble			LS Nibble
0	Torqrod 3	Torqrod 2	Torqrod 1

12. If only approximate values for the center of mass or moments of inertia are available, or these quantities are change only sufficiently slowly, they may be stored in the header block rather than the individual data blocks.

PL SETA Density Data Format
(continued)

Notes (continued):

13. The Eclipse Status will be designated by:
    - 0: no current eclipse
    - 1: penumbral eclipse phase
    - 2: umbral eclipse phase
14. The Calculation Option designations are:
    1) CALOPT = 0: (default) Compute densities and winds using only accelerometer data;
    2) CALOPT = 10: Compute densities and winds using ADMS mean molecular weights;
    3) CALOPT = 11: Compute winds using ADMS densities (and mean molecular weight);
    4) CALOPT = 20: Compute densities and winds using CADS mean molecular weights;
    5) CALOPT = 21: Compute densities and winds using CADS temperatures;
    6) CALOPT = 22: Compute densities and winds using CADS mean molecular weights and temperatures;
    7) CALOPT = 23: Compute winds using CADS densities (and mean molecular weight and temperature);
    8) CALOPT = 30: Compute winds using MSIS-90 densities.
15. The ADMS density and ADMS mean molecular weight items will be zero-filled if these are not available.
16. The CADS density, CADS mean molecular weight, CADS gas temperature, and CADS in-track wind items will be zero-filled if these are not available.
17. The Zero-order SETA Density is the value computed from the SETA Z-axis acceleration measurement only, with no adjustment for the in-track wind.

Glossary

ADMS	Absolute Density Mass Spectrometer
CADS	Composition and Density Sensor
CSTC	Consolidated Space Test Center
FIFO	First In, First Out
LS	Least Significant
MMHS	Mass Memory Header Structure
MMIT	Mass Memory Information Table
MS	Most Significant
MSIS	Mass Spectrometer and Incoherent Scatter
PSD	Power Spectral Density
SETA	Satellite Electrostatic Triaxial Accelerometer
SMC	Space and Missiles Center (incorporated CSTC)
TBD	"To Be Determined"
UT	Universal Time

SETA/ADS Software Development

APPENDIX A - Raw Data Unpacking Program

SETA/ADS Software Development
Raw Data Unpacking Program

```
Files:
     CSTC - SETA data in CSTC file format, after retrieval from
          tar archive
     ACCEL - SETA data in PL format
     LOG - log file for status reports

{INIT} [Initialization]
. Initialize variables:
. . INIT = TRUE [for output file header]
. . NSAMP = 0 [for number of samples in output block]
. . EXPDT = 0 [for expected composite SETA day and time of next
      sample group]
. Acquire user specifications:
. . CSTC input data file name;
. . ACCEL output data file name;
. . LOG listing file name;
. . Conversion factor for micro-G's to stored units (CVTSCL);
. Open CSTC data file for input [MS-F77:BINARY,
    VMS:UNFORMATTED/STREAM];
. Open ACCEL data file for output;
. Open LOG listing file for output;

{GET_CSTC} [Acquire data group from CSTC file]
. Read a CSTC data group (ten SETA samples, 76 bytes), swapping
    bytes as necessary;
. If error or end-of-file Then
. . Report error type and CSTC group number to LOG file;
. . Proceed to {INPEND};
. End if

{UNPACK} [Unpack the accelerometer data words, and store into PL
  format]
. Invoke {CVTGRP} with NSG = 10 to unpack and store the complete
    CSTC data group;
. Update expected composite SETA day and time for next group:
    EXPDT = CURDT + 1/86400.0
. Proceed from {GET_CSTC};

{INPEND} [Report end of CSTC data]
. If NSAMP > 0, then write current output block to ACCEL file:
. . DATDN, DATTIM, NSAMP, (ACCX(K), ACCY(K), ACCZ(K), TEMP(K),
      RNGFLG(K), K = 1, NSAMP)
. Close output file ACCEL;
. Report program conclusion;
```

Subroutines

```
{CVTGRP} [Convert input data group to output form, with proper
  blocking]
. Convert Time Tag Year/Month/Day to SETA day number (TMPDN);
. Convert Time Tag Hour/Minute/Second to time-of-day in tenths of
    seconds (TMPTIM), allowing for ~0.15 second offset;
. Unpack the ten SETA acceleration, range, and temperature values
    into individual words: (TMPX(J), TMPY(J), TMPZ(J), TMPT(J),
    TMPR(J), IRNGX(J), IRNGY(J), IRNGZ(J), J = 1, 10);[a]
. [Check for time gap in data, or full output block]
. Calculate composite SETA day/time:
. . CURDT = TMPDN + 0.1*TMPTIM/86400[b]
. Compare current date and time to previous date and time, for
    gap check:
. . GAP = (|CURDT - EXPDT| > 0.05/86400)[c]
. If GAP = TRUE or NSAMP ≥ 600 Then
. . If INIT = TRUE Then
. . . Assign values to output file header:
. . . . EXPID = 'SETA-5  '
. . . . DTYPE = 'RAW     '
. . . . SAMPRT = 10
. . . . DECIM = 1
. . . . ASCALE = 1/CVTSCL
. . . . BEGYR = TT_YR[d]
. . . . BEGMON = TT_MON
. . . . BEGDAY = TT_DAY
. . . . BEGDN = TMPDN
. . . . GENDN = SETA day number for processing date (from
          operating system)
. . . Write output file header, including 43 blank 4-byte words,
        to ACCEL file;
. . . Set INIT = FALSE;
. . End if
. . If NSAMP > 0 Then write current output block:
. . . DATDN, DATTIM, NSAMP, (ACCX(K), ACCY(K), ACCZ(K), TEMP(K),
        RNGFLG(K), K = 1, NSAMP)
. . Re-initialize sample counter: NSAMP = 0;
. . Store current SETA day and time for new output data header:
. . . DATDN = TMPDN
. . . DATTIM = TMPTIM
. End if
. [Store current samples for output]
. For J = 1 to NSG[e]
. . Increment sample counter: NSAMP = NSAMP + 1
. . [Convert temperatures from scaled counts to integer degrees
      Celsius]
. . TEMP(NSAMP) = CALT*TMPT(J)[f]
. . [Convert raw accelerometer counts to micro-gees, using a
      linear calibration based on the operating range and
```

```
     possibly also on the temperature, then to scaled counts]
. . ACCX(NSAMP) = CALX(IRNGX(J),TEMP(NSAMP))*TMPX(J)/ASCALE
. . ACCY(NSAMP) = CALY(IRNGY(J),TEMP(NSAMP))*TMPY(J)/ASCALE
. . ACCZ(NSAMP) = CALZ(IRNGZ(J),TEMP(NSAMP))*TMPZ(J)/ASCALE
. . RNGFLG(NSAMP) = TMPR(J)
. . Limit ACCX, ACCY, ACCZ to allowable range for two-byte
     integer;
. Next J
. Return to calling routine;
```

SETA/ADS Software Development
Raw Data Unpacking Program

Definitions and Notes

a. TMPX = temporary storage for x-acceleration
TMPY = temporary storage for y-acceleration
TMPZ = temporary storage for z-acceleration
TMPT = temporary storage for temperature
TMPR = temporary storage for packed range flags
IRNGX = numerical equivalent for accelerometer x-axis range
IRNGY = numerical equivalent for accelerometer y-axis range
IRNGZ = numerical equivalent for accelerometer z-axis range

b. CURDT = current composite SETA day and time, as day and fraction (double precision)

c. GAP = logical variable, set TRUE for time gap

d. TT_YR = Time Tag Year
TT_MON = Time Tag Month
TT_DAY = Time Tag Day of Month

e. NSG = number of samples in a one-second CSTC data group (up to 10)

f. CALX = calibration coefficient for accelerometer x-axis, for counts to micro-gees
CALY = calibration coefficient for accelerometer y-axis, for counts to micro-gees
CALZ = calibration coefficient for accelerometer z-axis, for counts to micro-gees
CALT = calibration coefficient for temperature counts to degrees Celsius

SETA/ADS Software Development
Raw Data Unpacking Program

APPENDIX B - Raw Data Checking Program

SETA/ADS Software Development
Raw Data Checking Program

Files:
CSTC - SETA data in CSTC file format, after retrieval from <u>tar</u> archive
LIST - listing file of times, averages, and standard deviations
LOG - log file of data events (range changes, saturations, temperature excursions)
ACCEL - SETA data in PL format

{OPTS} [Obtain processing options from user]
. Read processing options from file or terminal, or retain defaults, in parentheses:
. . GENOUT (= FALSE) [generate standard ACCEL file if TRUE]
. . LISTSTAT (= TRUE) [generate listing of 2-minute averages and standard deviations for accelerations and temperature]
. . DSTART(3) (= 12,31,1989) [calendar month, day, and year for selecting start of data]
. . DSTOP(3) (= 12,31,2001) [calendar month, day, and year for selecting end of data]
. . TSTART (= 0) [start time in seconds for selecting data]
. . TSTOP (= 86400) [stop time in seconds for selecting data]
. . SATHI (= 2047) [upper limit for accelerometer saturation count]
. . SATLO (= -2048) [lower limit for accelerometer saturation count]
. . TEMPHI (= 50) [upper limit for accelerometer operating temperature, in degrees Celsius]
. . TEMPLO (= 0) [lower limit for accelerometer operating temperature, in degrees Celsius]

{INIT} [Initialization]
. Initialize variables:
. . INIT = TRUE [for output file header]
. . PROC = FALSE [for processing current time samples]
. . TERM = FALSE [for termination of processing]
. . NSAMP = 0 [for number of samples in output block]
. . EXPDT = 0 [for expected composite SETA day and time of next sample group]
. . NAVG = 0 [number of samples in current averaging group]
. . AVGX = 0 [X-acceleration average for current averaging group]
. . AVGY = 0 [Y-acceleration average for current averaging group]
. . AVGZ = 0 [Z-acceleration average for current averaging group]
. . AVGT = 0 [temperature average for current averaging group]
. . STDVX = 0 [X-acceleration standard deviation for current averaging group]
. . STDVY = 0 [Y-acceleration standard deviation for current averaging group]
. . STDVZ = 0 [Z-acceleration standard deviation for current averaging group]

```
. . . STDVT = 0 [temperature standard deviation for current
        averaging group]
. . TIMEAVG = 0 [composite SETA day and time for start of current
      averaging group]
. . PRNGX = -1 [reference X-acceleration range for comparison to
      current sample]
. . PRNGY = -1 [reference Y-acceleration range for comparison to
      current sample]
. . PRNGZ = -1 [reference Z-acceleration range for comparison to
      current sample]
. Open CSTC data file for input [MS-F77:BINARY,
    VMS:UNFORMATTED/STREAM];
. If GENOUT = TRUE Then open ACCEL data file for output;
. If LISTOUT = TRUE Then open LIST listing file for output;
. Open LOG listing file for output;
. Convert DSTART and TSTART to composite SETA day and time
    STARTDT for start of selected data segment;
. Convert DSTOP and TSTOP to composite SETA day and time STOPDT
    for end of selected data segment;

{GET_CSTC} [Acquire data group from CSTC file]
. Read a CSTC data group (ten SETA samples, 76 bytes), swapping
    bytes as necessary;
. If error or end-of-file Then
. . Report error type and CSTC group number to LOG file;
. . Proceed to {INPEND};
. End if

{UNPACK} [Unpack the accelerometer data words, and store into PL
  format]
. Invoke {CVTGRP} with NSG = 10 to unpack and (conditionally)
    store the complete CSTC data group;
. If PROC = FALSE Then proceed from {GET_CSTC};
. Update expected composite SETA day and time for next group:
    EXPDT = CURDT + 1/86400.0

{CHECK} [Check the current data group for anomalies]
. For I = 1 to 10
. . If (TMPX(I) or TMPY(I) or TMPZ(I)) ≥ SATHI or (TMPX(I) or
      TMPY(I) or TMPZ(I)) ≤ SATLO Then report SETA day, time, and
      acceleration count, range, and axis to LOG file;
. . If TEMP(NSAMP-10+I) ≥ TEMPHI or TEMP(NSAMP-10+I) ≤ TEMPLO
      Then report SETA day, time, and temperature to LOG file;
. . If IRNGX(I) ≠ PRNGX or IRNGY(I) ≠ PRNGY or IRNGZ(I) ≠ PRNGZ
      Then report SETA day, time, and accelerometer range pair
      and axis to LOG file;
. . Update the values for the reference accelerometer ranges:
. . . PRNGX = IRNGX(I);
. . . PRNGY = IRNGY(I);
. . . PRNGZ = IRNGZ(I);
```

```
. . . If LISTOUT = TRUE Then
. . . If CURDT < TIMEAVG + 120/86400.0 and TERM = FALSE Then
. . . . [Continue to accumulate statistics]
. . . . AVGX = AVGX + ACCX(NSAMP-10+I);
. . . . AVGY = AVGY + ACCY(NSAMP-10+I);
. . . . AVGZ = AVGZ + ACCZ(NSAMP-10+I);
. . . . AVGT = AVGT + ACCT(NSAMP-10+I);
. . . . STDVX = STDVX + ACCX(NSAMP-10+I)^2;
. . . . STDVY = STDVY + ACCY(NSAMP-10+I)^2;
. . . . STDVZ = STDVZ + ACCZ(NSAMP-10+I)^2;
. . . . STDVT = STDVT + ACCT(NSAMP-10+I)^2;
. . . . NAVG = NAVG + 1;
. . . Else
. . . . [Report averaging group values (even if only partial
         group on termination) and re-initialize statistics]
. . . . If NAVG > 0 Then
. . . . . AVGX = AVGX/NAVG;
. . . . . AVGY = AVGX/NAVG;
. . . . . AVGZ = AVGX/NAVG;
. . . . . AVGT = AVGX/NAVG;
. . . . . STDVX = √(STDVX/NAVG - AVGX^2);
. . . . . STDVY = √(STDVY/NAVG - AVGY^2);
. . . . . STDVZ = √(STDVZ/NAVG - AVGZ^2);
. . . . . STDVT = √(STDVT/NAVG - AVGT^2);
. . . . . Report calendar date, current time in
            hours/minutes/seconds, and NAVG, AVGX, STDVX, AVGY,
            STDVY, AVGZ, STDVZ, AVGT, STDVT to LIST file;
. . . . End if
. . . . AVGX = ACCX(NSAMP-10+I);
. . . . AVGY = ACCY(NSAMP-10+I);
. . . . AVGZ = ACCZ(NSAMP-10+I);
. . . . AVGT = ACCT(NSAMP-10+I);
. . . . STDVX = ACCX(NSAMP-10+I)^2;
. . . . STDVY = ACCY(NSAMP-10+I)^2;
. . . . STDVZ = ACCZ(NSAMP-10+I)^2;
. . . . STDVT = ACCT(NSAMP-10+I)^2;
. . . . NAVG = 1;
. . . . TIMEAVG = CURDT;
. . . End if
. . End if
. Next I
. If TERM = FALSE Then proceed from {GET_CSTC};

{INPEND} [Report end of CSTC data]
. If GENOUT = TRUE Then
. . If NSAMP > 0 Then write current output block to ACCEL file:
. . . DATDN, DATTIM, NSAMP, (ACCX(K), ACCY(K), ACCZ(K), TEMP(K),
        RNGFLG(K), K = 1, NSAMP)
. . Close output file ACCEL;
. End if
```

```
. If LISTOUT = TRUE Then close listing file LIST;
. Close log file LOG;
. Report program conclusion;
```

Subroutines

```
{CVTGRP} [Convert input data group to output form, with proper
  blocking]
. Convert Time Tag Year/Month/Day to SETA day number (TMPDN);
. Convert Time Tag Hour/Minute/Second to time-of-day in tenths of
    seconds (TMPTIM), allowing for ~0.15 second offset;
. Unpack the ten SETA acceleration, range, and temperature values
    into individual words: (TMPX(J), TMPY(J), TMPZ(J), TMPT(J),
    TMPR(J), IRNGX(J), IRNGY(J), IRNGZ(J), J = 1, 10);(a)
. [Check for time gap in data, or full output block]
. Calculate composite SETA day and time:
. . CURDT = TMPDN + 0.1*TMPTIM/86400(b)
. [Check current time against start and stop times]
. If CURDT < STARTDT Then return to calling routine;
. Set PROC = TRUE to enable processing;
. If CURDT > STOPDT Then
. . Set TERM = TRUE to terminate processing;
. . Return to calling routine;
. End if
. Compare current date and time to previous date and time, for
    gap check:
. . GAP = (|CURDT - EXPDT| > 0.05/86400)(c)
. If GAP = TRUE Then report beginning and ending time of gap to
    LOG file;
. If GENOUT = TRUE and (GAP = TRUE or NSAMP ≥ 600) Then
. . If INIT = TRUE Then
. . . Assign values to output file header:
. . . . EXPID = 'SETA-5   '
. . . . DTYPE = 'RAW      '
. . . . SAMPRT = 10
. . . . DECIM = 1
. . . . ASCALE = 0.01
. . . . BEGYR = TT_YR(d)
. . . . BEGMON = TT_MON
. . . . BEGDAY = TT_DAY
. . . . BEGDN = TMPDN
. . . . GENDN = SETA day number for processing date (from
          operating system)
. . . Write output file header, including 43 blank 4-byte words,
        to ACCEL file;
. . . Set INIT = FALSE;
. . End if
. . If NSAMP > 0 Then write current output block:
. . . DATDN, DATTIM, NSAMP, (ACCX(K), ACCY(K), ACCZ(K), TEMP(K),
```

```
        RNGFLG(K), K = 1, NSAMP)
. . Re-initialize sample counter: NSAMP = 0;
. . Store current SETA day and time for new output data header:
. . . DATDN = TMPDN
. . . DATTIM = TMPTIM
. End if
. [Store current samples for output]
. For J = 1 to NSG(e)
. . Increment sample counter: NSAMP = NSAMP + 1
. . [Convert temperatures from scaled counts to integer degrees
    Celsius]
. . TEMP(NSAMP) = CALT*TMPT(J)(f)
. . [Convert raw accelerometer counts to micro-gees, using a
    linear calibration based on the operating range and
    possibly also on the temperature, then to scaled counts]
. . ACCX(NSAMP) = CALX(IRNGX(J))*TMPX(J)/ASCALE
. . ACCY(NSAMP) = CALY(IRNGY(J))*TMPY(J)/ASCALE
. . ACCZ(NSAMP) = CALZ(IRNGZ(J))*TMPZ(J)/ASCALE
. . RNGFLG(NSAMP) = TMPR(J)
. Next J
. Return to calling routine;
```

SETA/ADS Software Development
Raw Data Checking Program

Definitions and Notes

a. TMPX = temporary storage for x-acceleration
TMPY = temporary storage for y-acceleration
TMPZ = temporary storage for z-acceleration
TMPT = temporary storage for temperature
TMPR = temporary storage for packed range flags
IRNGX = numerical equivalent for accelerometer x-axis range
IRNGY = numerical equivalent for accelerometer y-axis range
IRNGZ = numerical equivalent for accelerometer z-axis range

b. CURDT = current composite SETA day and time, as day and fraction (double precision)

c. GAP = logical variable, set TRUE for time gap

d. TT_YR = Time Tag Year
TT_MON = Time Tag Month
TT_DAY = Time Tag Day of Month

e. NSG = number of samples in a one-second CSTC data group (up to 10)

f. CALX = calibration coefficient for accelerometer x-axis, for counts to micro-gees
CALY = calibration coefficient for accelerometer y-axis, for counts to micro-gees
CALZ = calibration coefficient for accelerometer z-axis, for counts to micro-gees
CALT = calibration coefficient for temperature counts to degrees Celsius

SETA/ADS Software Development
Raw Data Checking Program

APPENDIX C - Power Spectral Density Program

SETA/ADS Software Development
Power Spectral Density Program

Files:
DATA - SETA raw or filtered accelerometer data in PL format

Parameters:
MAXSMP = 600 [maximum number of samples in a data block (DATA)]
LENBUF = 4096 [length of buffer (array) for storage of data samples to be analyzed]

Overview:
1. Acquire data values from DATA into interim storage (ACCDATA, FLAGVAL), with checking for time gaps;
2. Transfer data to processing buffers (BUF), for "wild point" checking (if enabled) and Fourier transform, with interpolation across removable gaps;
3. Perform the Fourier transform for the specified acceleration or temperature values, storing the PSD amplitudes (PSDAMP);
4. Determine the plot frame specifications according to the user requirements, and plot the PSD.

```
{INIT} [Initialization]
. Open DATA data file for input;
. Open SPEC parameter file for input;
. Initialize variables:
. . TLAST = 0 [composite day number/time for last sample in
    buffer]
. . TNEXT = 0 [composite day number/time for next sample expected
    for buffer]
. . TSTART = 0 [composite day number/time for ISTART sample in
    buffer]
. . TFIRST = 0 [composite day number/time for first sample in
    buffer after requested start time]
. . STAT = blank (' ') [status for data acquisition]

{READ_SPEC} [Read processing specifications from user](a)
. Read user specifications for each data sequence (accelerations
   and temperature), with defaults in angle brackets:
. . PSDCAL(4) <TRUE,TRUE,TRUE,FALSE> [Perform PSD calculation]
. . DNBEG [Beginning SETA day number for selection of data
    samples]
. . DTBEG [Beginning time-of-day for selection of data samples,
    in seconds]
. . NTPSD [Number of time samples to be used for PSD]
. . WTYPE(4) <'        ','        ','        ','        '>
    [Windowing type, defaulting to none]
. . MWNDW(4) <(TBD)> [Median window length, in samples, for "wild
    point" removal]
. . WPTHR(4) <(TBD)> [Threshold level, in micro-Gees or degrees
    Celsius, for "wild point" removal]
```

. . WPEDIT(4) <'Y','Y','Y','Y'> [Flags for "wild point" editing]
. . PSDGAP <**(TBD)**> [Gap threshold, in seconds]
. . LOGAMP <TRUE> [Plot PSD amplitudes on logarithmic scale; alternative is linear scale]
. . AUTOSC <TRUE> [Determine ordinate range based on data range (autoscale)]
. . FRMAX <0.0, 0.0, 0.0, 0.0> [Maximum frequency for plot - default values trigger auto-scaling][b]
. . AMPMAX <10.0, 10.0, 10.0, 100.0> [Maximum PSD amplitude for plot, in units corresponding to LOGAMP]
. . AMPMIN <0.0, 0.0, 0.0, 0.0> [Minimum PSD amplitude for plot, in units corresponding to LOGAMP; also used as threshold for conversion to logarithms][c]
. . PDATE <FALSE> [Flag to include plot generation date in caption]

Notes:
1) It would also be possible to specify a minimal data segment length, in seconds, but this option will be excluded unless justified by many data sequences.

{READ_HDR} [Read and store header information for input file]
. Read RAW DATA or FILTER header items from file DATA: [See data format descriptions]
. . EXPID
. . DTYPE
. . SAMPRT
. . DECIM
. . ASCALE
. . BEGYR
. . BEGMON
. . BEGDAY
. . BEGDN
. . GENDN
. . FILTDN (possibly blank)
. . TGAP (possibly blank)
. . WPTHR(1) (X, possibly blank)
. . WPTHR(2) (Y, possibly blank)
. . WPTHR(3) (Z, possibly blank)
. . WPTHR(4) (T, possibly blank)
. . WPED(1) (X, possibly blank)
. . WPED(2) (Y, possibly blank)
. . WPED(3) (Z, possibly blank)
. . WPED(4) (T, possibly blank)
. . FILT(K,1), K = 1, 4 (X, possibly blank)
. . FILT(K,2), K = 1, 4 (Y, possibly blank)
. . FILT(K,3), K = 1, 4 (Z, possibly blank)
. . FILT(K,4), K = 1, 4 (T, possibly blank)
. [Initialize header-dependent variables]
. TINT = DECIM/SAMPRT [time interval between samples, in seconds]
. TINC = TINT/86400.0 [composite day number/time for sampling

```
    interval]

{CHK_SPEC} [Check compatibility of specifications, and define
  supplementary variables]
. [Number of time samples must be a power of two, and not greater
    than LENBUF]
. If NTPSD > LENBUF Then
. . Warn user of NTPSD setting (larger than buffer length) and
      reassignment;
. . Reset NTPSD = LENBUF;
. End if
. NTEXP = ROUND(ln(NTPSD)/ln(2))
. NTSET = 2^NTEXP
. If NTSET ≠ NTPSD Then
. . Warn user of NTPSD setting (not an integral power of two) and
      reassignment;
. . NTPSD = NTSET
. End if
. [Check allowable gap threshold against (decimated) sampling
    interval]
. If PSDGAP ≤ TINT Then
. . Warn user of (PSDGAP, TINT) inconsistency, and reassignment;
. . PSDGAP = 1.5*TINT
. End if
. [Check median window length against number of samples
    requested]
. For I = 1 to 4
. . If WPEDIT(I) = 'Y' and MWNDW(I) > NTPSD Then
. . . Warn user of MWNDW(I), NTPSD inconsistency, and
        reassignment, for index I;
. . . MWNDW(I) = NTPSD
. . End if
. Next I
. [Set beginning time in day and fraction format]
. TBEGIN = DNBEG + DTBEG/86400.0

{START_BUF} [Initialize the buffer with data samples started at
  the time requested for PSD]
. While TLAST < TBEGIN and STAT ≠ 'DONE'
. . ISTART = 1(d)
. . Invoke LD_BUF(BUF, LENBUF, ASCALE, TSTART, TLAST, TNEXT,
      TINC, PSDGAP, ISTART, IEND, MBEG, MEND, STAT) to load
      buffers for each data sequence;
. End while
. If STAT = 'DONE' Then proceed to {END_PSD};
. [Determine the sample index corresponding to the requested
    beginning time]
. If TBEGIN > TSTART Then
. . [The requested beginning time lies somewhere in the buffer,
      so find the corresponding index]
```

```
. . ISAMP1 = 1 + CEILING((TBEGIN - TSTART)/TINC)(e)
. . TFIRST = TSTART + (ISAMP1 - 1)*TINC
. Else
. . [The requested beginning time lies in a gap prior to TSTART,
      but possibly only due to the sampling interval]
. . ISAMP1 = 1
. . TFIRST = TSTART
. Endif

{SHIFT1} [Shift the first selected and subsequent samples to the
  beginning of the buffer, and assign index for next segment
  acquisition]
. If ISAMP1 > 1 Then
. . For I = 1 to 4
. . . K = 1
. . . For J = ISAMP1 to IEND
. . . . BUF(K,I) = BUF(J,I)
. . . . K = K + 1
. . . Next J
. . Next I
. . ISTART = K
. . IEND = ISTART - 1
. Else
. . [No shifting required]
. . ISTART = IEND + 1
. End if

{REPL_WILD} [Discover "wild points" in acquired segment, and
  replace by local median]
. For I = 1 to 4
. . If WPEDIT(I) = 'Y' Then
. . . Invoke WP_CHECK(BUF, I, 1, IEND, MWNDW(I), WPTHR(I)) to
        edit "wild points" for selected data segment;
. . End if
. Next I

{FILL_BUF} [Continue filling the buffer until the requested
  number of samples is obtained (counting removable gaps), or
  until a non-removable gap is encountered]
. While IEND < NTPSD and (STAT ≠ 'TERM' and STAT ≠ 'DONE')
. . Invoke LD_BUF(BUF, LENBUF, ASCALE, TSTART, TLAST, TNEXT,
      TINC, PSDGAP, ISTART, IEND, MBEG, MEND, STAT) to load
      buffers for each data sequence;
. . [Discover "wild points" in acquired segment, and replace by
      local median]
. . If STAT ≠ 'DONE' and STAT ≠ 'TERM' Then
. . . [Some new data were retrieved, so check for "wild points"]
. . . For I = 1 to 4
. . . . If WPEDIT(I) = 'Y' Then
. . . . . If STAT = 'FILL' Then
```

```
. . . . . . Invoke WP_CHECK(BUF, I, MEND+1, IEND, MWNDW(I),
              WPTHR(I)) to edit "wild points" for acquired data
              segment;
. . . . . Else
. . . . . . Invoke WP_CHECK(BUF, I, ISTART, IEND, MWNDW(I),
              WPTHR(I)) to edit "wild points" for acquired data
              segment;
. . . . . End if
. . . . End if
. . . Next I
. . End if
. . If STAT = 'FILL' Then
. . . Report that interpolation is being performed across a
        removable gap, with the associated values TSTART, MEND,
        NMISS = MEND - MBEG + 1;(f)
. . . {INTERP} [Interpolate between two samples on either side of
        a removable gap]
. . . For I = 1 to 4
. . . . DSTEP = (BUF(MEND+1,I) - BUF(MBEG-1,I))/(MEND - MBEG + 2)
. . . . For J = MBEG to MEND
. . . . . BUF(J,I) = BUF(J-1,I) + DSTEP
. . . . Next J
. . . Next I
. . End if
. . [Assign index for next segment acquisition]
. . ISTART = IEND + 1
. End while

{SAMP_RPT} [Define the number of samples to reflect the number
  actually acquired, adjusted for an integral power of two]
. [Must have IEND > 0 here, or initial LD_BUF invocation would
    have reported 'DONE', bypassing these steps]
. NSPSD = Min(IEND, NTPSD)
. NTEXP = INT(ln(NSPSD)/ln(2))
. NSPSD = 2^NTEXP
. If NSPSD ≠ NTPSD Then warn user that actual PSD size is
    different from requested size, reporting values of NSPSD and
    NTPSD (not enough points acquired before gap);

{BLD_WIND} [Calculate and apply the windowing weights for the
  associated sample count duration, based on specified parameters
  (one set each for X-acceleration, Y-acceleration, Z-
  acceleration, and temperature)]
. For I = 1 to 4
. . If PSDCAL(I) is TRUE Then
. . . WTSQ = 0.0(g)
. . . If WTYPE(I) = 'HANNING ' Then
. . . . [Impose a Hanning window]
. . . . For K = 1 to NSPSD
. . . . . WT = 0.5*(1 - cos(2*π*(K-1)/NSPSD))
```

```
. . . . . . BUF(K,I) = WT*BUF(K,I)
. . . . . . WTSQ = WTSQ + WT**2
. . . . . Next K
. . . . . PSDFCT = TINT/WTSQ
. . . . Else If WTYPE(I) = 'HAMMING ' Then
. . . . . [Impose a Hamming window]
. . . . . For K = 1 to NSPSD
. . . . . . WT = 0.54 - 0.46*cos(2*π*(K-1)/NSPSD)
. . . . . . BUF(K,I) = WT*BUF(K,I)
. . . . . . WTSQ = WTSQ + WT**2
. . . . . Next K
. . . . . PSDFCT = TINT/WTSQ
. . . . Else If WTYPE(I) = 'BLACKMAN' Then
. . . . . [Impose a Blackman window]
. . . . . For K = 1 to NSPSD
. . . . . . WT = 0.42 - 0.5*cos(2*π*(K-1)/NSPSD) +
              0.08*cos(4*π*(K-1)/NSPSD)
. . . . . . BUF(K,I) = WT*BUF(K,I)
. . . . . . WTSQ = WTSQ + WT**2
. . . . . Next K
. . . . . PSDFCT = TINT/WTSQ
. . . . Else
. . . . . [No windowing (equivalent to rectangular window) imposed,
            if no strings match]
. . . . . PSDFCT = TINT/NSPSD
. . . . End if
. . . . {CALC_PSD} [Calculate the PSD for the requested data
          sequences]
. . . . Invoke PSD(BUF(1,I),NSPSD,PSDFCT,PSDAMP,NAMPL) to calculate
          the PSD amplitudes, PSDAMP(K), K = 1, NAMPL;
. . . . Invoke PLOT_PSD(I, PSDAMP, NAMPL, TFIRST, LOGAMP, AUTOSC,
          FRMAX(I), AMPMAX(I), AMPMIN(I), PDATE, TINT, DTYPE,
          GENDN, FILTDN, FILT(1,I)) to plot the PSD amplitudes;
. . . End if
. Next I

{END_PSD} [Conclude processing]
. Report program conclusion;
```

Subroutines

LD_BUF(BUF, LENBUF, ASCALE, TSTART, TLAST, TNEXT, TINC, TYMGAP, ISTART, IEND, MBEG, MEND, STAT)

- BUF(LENBUF,4) = (R*4) array for individual data sequences (X,Y,Z,T) [output];
- LENBUF = (I*2) time-sequence dimension for BUF (maximum number of samples) [input];
- ASCALE = (R*4) scale factor for stored acceleration counts to micro-Gees [input];
- TSTART = (R*8) composite day/time for beginning of current data group acquired from DATA [output];
- TLAST = (R*8) composite day number/time for last sample in filtering buffer [output];
- TNEXT = (R*8) composite day number/time for next sample expected for filtering buffer [input/output];
- TINC = (R*8) composite day number/time for sampling interval [input];
- TYMGAP = (R*4) gap threshold, in seconds [input];
- ISTART = (I*2) initial index at which to start storing data [input];
- IEND = (I*2) index at which data storage ends (for end of BUF or time gap, including end-of-data) [output];
- MBEG = (I*2) initial index at which replaceable missing data occurs [output];
- MEND = (I*2) last index at which replaceable missing data occurs [output];
- STAT = (C*4) status of data acquisition [output]:
  - 'INIT' = data acquired after time gap;
  - 'OKAY' = data acquired with no immediately preceding time gap;
  - 'TERM' = data ends at time gap;
  - 'FILL' = data segment contains removable time gap;
  - 'DONE' = all input values have been acquired;

Local variables:

- DATDN = (I*2) SETA day number for beginning of data block
- DATTIM = (I*4) time of day for beginning of data block, in tenths of seconds
- ACCDATA(MAXSMP,4) = (I*2) scaled accelerations and temperature
- FLAGVAL(MAXSMP) = (I*2) packed range flags

Initialization values:

- IOSTAT = 0 [file read return status]
- NREM = 0 [number of samples remaining for transfer to BUF]
- NSAMP = 0 [number of samples in data group acquired from DATA]
- KREM = 1 [index of first sample remaining after incomplete transfer to BUF]

```
. If IOSTAT = -1 Then
. . Set STAT = 'DONE'
. . Return to calling routine;
```

```
. End if
. {FETCH} Set IOSTAT = 0 [initialize for successful read];
. If NREM = 0 Then
. . Read DATDN, DATTIM, NSAMP, ((ACCDATA(K,L), L = 1, 4),
      FLAGVAL(K), K = 1, NSAMP) from DATA;
. . If end-of-file on read Then
. . . [This should not happen during a data block without error];
. . . Set IOSTAT = -1; [standard FORTRAN result]
. . . Set STAT = 'TERM';
. . . Return to calling routine;
. . Else if error on read Then
. . . Set IOSTAT = error number;
. . . Report error in data acquisition;
. . . Set STAT = 'DONE' [note difference from FILTER LOAD_BUF]
. . . Return to calling routine;
. . End if
. . CURRDT = DATDN + DATTIM/86400.0
. End if
. [Compare current initial day/time for block to expected
    day/time]
. If |CURRDT - TNEXT| > 0.5*TINC Then
. . [A time gap exists (or the data sequence has just begun)]
. . If |CURRDT - TNEXT| > TYMGAP/86400.0 Then
. . . [This is a permanent gap]
. . . If TLAST = 0 Then
. . . . Set STAT = 'INIT';
. . . . IEND = Min(ISTART+NSAMP-1, LENBUF) [note difference from
          FILTER LOAD_BUF]
. . . . NREM = NSAMP - (IEND - ISTART) - 1
. . . . [Load processing buffer]
. . . . For I = 1 to 3
. . . . . K = KREM
. . . . . For J = ISTART to IEND
. . . . . . BUF(J,I) = ASCALE*ACCDATA(K,I)
. . . . . . K = K + 1
. . . . . Next J
. . . . Next I
. . . . K = KREM
. . . . For J = ISTART to IEND
. . . . . BUF(J,4) = ACCDATA(K,4)
. . . . . K = K + 1
. . . . Next J
. . . . MBEG = IEND
. . . . MEND = IEND
. . . Else
. . . . Set STAT = 'TERM';
. . . . IEND = ISTART - 1
. . . . NREM = NSAMP
. . . . KREM = 1
. . . . TLAST = 0
```

```
. . . . . Return to calling routine;
. . . . End if
. . . Else [This is a removable time gap]
. . . . Set STAT = 'FILL';
. . . . MBEG = ISTART
. . . . NMISS = Round((CURRDT - TNEXT)/TINC) [must be at least one,
          by original test condition]
. . . . MEND = ISTART + NMISS - 1
. . . . [Insure that removable gap does not straddle upper index
          limit of buffer, thus impeding interpolations]
. . . . If MEND ≥ LENBUF Then
. . . . . [Treat this as a permanent gap; note difference from
            FILTER LOAD_BUF]
. . . . . Set STAT = 'TERM';
. . . . . IEND = ISTART - 1
. . . . . NREM = NSAMP
. . . . . KREM = 1
. . . . . TLAST = 0
. . . . . Return to calling routine;
. . . . End if
. . . . IEND = Min(MEND+NSAMP, LENBUF) [note difference from FILTER
          LOAD_BUF]
. . . . NREM = NSAMP - (IEND - MEND)
. . . . [Load processing buffer]
. . . . For I = 1 to 3
. . . . . K = KREM
. . . . . For J = MEND+1 to IEND
. . . . . . BUF(J,I) = ASCALE*ACCDATA(K,I)
. . . . . . K = K + 1
. . . . . Next J
. . . . Next I
. . . . K = KREM
. . . . For J = MEND+1 to IEND
. . . . . BUF(J,4) = ACCDATA(K,4)
. . . . . K = K + 1
. . . . Next J
. . . End if
. . Else
. . . [No time gap]
. . . Set STAT = 'OKAY';
. . . IEND = Min(ISTART+NSAMP-1, LENBUF) [note difference from
        FILTER LOAD_BUF]
. . . NREM = NSAMP - (IEND - ISTART) - 1
. . . [Load processing buffer]
. . . For I = 1 to 3
. . . . K = KREM
. . . . For J = ISTART to IEND
. . . . . BUF(J,I) = ASCALE*ACCDATA(K,I)
. . . . . K = K + 1
. . . . Next J
```

```
. . Next I
. . K = KREM
. . For J = ISTART to IEND
. . . BUF(J,4) = ACCDATA(K,4)
. . . K = K + 1
. . Next J
. . MBEG = IEND
. . MEND = IEND
. End if
. TSTART = CURRDT
. TLAST = CURRDT + (NSAMP - NREM - 1)*TINC [time of last sample
    in BUF]
. TNEXT = TLAST + TINC
. CURRDT = TNEXT [update current time for remaining samples]
. KREM = Mod(NSAMP - NREM + 1, NSAMP) [first index of remaining
    samples]
. NSAMP = NREM
. Return to calling routine;
```

```
WP_CHECK(BUF, I, ISTART, IEND, MWNDW, WPTHR)
  BUF = array for individual data sequence [input/output];
  I = selection index for data type (acceleration or temperature)
  ISTART = index of first sample to be checked;
  IEND = index of last sample to be checked;
  MWNDW = number of samples for median referencing;
  WPTHR = threshold level for wild point exclusion;
  [see existing EDITDTA routine, but use comparison to WPTHR
    rather than local standard deviations]
  [also check condition MWNDW > IEND - ISTART + 1]
  [note: no flag bit settings are required for the data flags]
```

```
PSD(SEQ,NSPSD,PSDFCT,PSDAMP,NAMPL)
  SEQ(NSPSD) = (R*4) array for time-sequence data, with desired
    windowing factors applied [input];
  NSPSD = (I*2) number of time samples to use for power spectrum
    [input];
  PSDFCT = (R*4) normalization factor for PSD amplitudes [input];
  PSDAMP = (R*4) array for PSD amplitudes [output];
  NAMPL = (I*2) actual number of PSD amplitudes returned
    [output];
Local variables:
  NFFT = (I*2) number of elements designated for the Fourier
    transform routine;
Initialization values:
  ISIGN = 1 [designates forward Fourier transform]

{FFT} [Perform Fast Fourier Transform in place on original time-
  sequence data]
```

```
. NFFT = NSPSD/2
. Invoke REALFT(SEQ,NFFT,ISIGN) [Numerical Recipes - NRFFT.FOR]
    to calculate the positive-frequency FFT coefficients for the
    designated time sequence data;

{CALC_AMP} [Compute the amplitudes, with the appropriate mapping
  and normalization]
. PSDAMP(1) = SEQ(1)^2*PSDFCT
. NAMPL = NFFT + 1
. PSDAMP(NAMPL) = SEQ(2)^2*PSDFCT
. K = 2
. For J = 3 to NSPSD-1, by steps of 2
. . PSDAMP(K) = (SEQ(J)^2 + SEQ(J+1)^2)*PSDFCT
. . K = K + 1
. Next J

. Return to calling routine;
```

```
PLOT_PSD(ITYPE, PSDAMP, NAMPL, TFIRST, LOGAMP, AUTOSC, FRMAX,
  AMPMAX, AMPMIN, PDATE, TINT, DTYPE, GENDN, FILTDN, FILT)
  ITYPE = (I*2) index for data sequence (accelerations or
    temperature) [input];
  PSDAMP = (R*4) array for PSD amplitudes [input];
  NAMPL = (I*2) number of PSD amplitudes [input];
  TFIRST = (R*8) composite day number/time for first time sample
    used for PSD [input];
  LOGAMP = (L*2) flag for plotting PSD amplitudes on logarithmic
    scale [input];
  AUTOSC = (L*2) flag for invoking auto-scaling for ordinate (PSD
    amplitudes) [input];
  FRMAX = (R*4) maximum frequency, in Hz, for plot, triggering
    auto-scaling if zero [input];
  AMPMAX = (R*4) maximum PSD amplitude for plot (linear or
    logarithmic value) [input];
  AMPMIN = (R*4) minimum PSD amplitude for plot (linear or
    logarithmic value) [input];
  PDATE = (L*2) flag to include plot generation date in caption
    [input];
  TINT = (R*4) time interval between data samples, in seconds
    [input];
  DTYPE = (C*8) data type for source of data ('RAW     ' or
    'FILTER  ') [input];
  GENDN = (I*2) SETA day number for Raw Data processing [input];
  FILTDN = (I*2) SETA day number for Filter processing [input];
  FILT(4,4) = (R*4) Filtering parameters, from data source
    [input];
Local variables:
  FRQVAL(BUF_LEN) = (R*4) frequency values associated with each
    PSD amplitude
```

```
Initialization values:
  FRMIN = 0.0 [lower limit for plotted frequency range, in Hz]

{FREQ} [Calculate the frequency values for the PSD, based on the
  sampling interval and time sequence duration]
. FRQLIM = 0.5/TINT
. FRQINC = FRQLIM/(NAMPL-1)
. [Determine the maximum index to be used for plotting, based on
    the requested plot limits]
. If FRMAX = 0.0 Then
. . KMAX = NAMPL
. Else
. . KMAX = FRMAX/FRQINC + 1
. End if
. [Define the frequencies]
. For K = 1 to KMAX
. . FRQVAL(K) = (K - 1)*FRQINC
. Next K

{CVTLOG} [If required, convert the amplitudes to logarithms]
. If LOGAMP is TRUE Then
. . AMPREF = 10^AMPMIN
. . For K = 1 to KMAX
. . . If PSDAMP(K) > AMPREF Then
. . . . PSDAMP(K) = LOG_10(PSDAMP(K))
. . . Else
. . . . PSDAMP(K) = AMPMIN
. . . End if
. . Next K
. End if

. If AUTOSC is TRUE Then
. . Set AMPMAX = MAX{PSDAMP(K); K = 1, KMAX};
. . Set AMPMIN = MIN{PSDAMP(K); K = 1, KMAX};
. Else
.   Truncate PSDAMP amplitudes to lie within (AMPMAX, AMPMIN);
    [this is partially done if LOGAMP is TRUE, using the
    threshold provisions for the conversion to logarithms]
. End if

. Define plot axes for abscissa range (FRMIN, FRMAX) and ordinate
    range (AMPMIN, AMPMAX);

. Plot PSDAMP versus FRQVAL for K = 1, KMAX;

. Label plot with calendar date (dd-mon-year) for data, derived
    from TFIRST;(h)
. Label plot with time of day, in hours, minutes, and seconds,
    derived from TFIRST;
. Label plot with data type (X-acceleration, Y-acceleration,
```

```
   Z-acceleration, temperature), based on the ITYPE value;
. Write caption for plot, including the following items:
. . Duration of data segment, as number of samples (NSPSD) and
      total time interval in seconds ((NSPSD-1)*TINT);
. . Date of Raw Data processing, as calendar date derived from
      GENDN;
. . If DTYPE = 'FILTER  ':
. . . Date of Filter processing, as calendar date derived from
        FILTDN;
. . . If FILT(1,ITYPE) = 'L':
. . . . "Low-pass Filter"
. . . . "Passband:" = FILT(2,ITYPE) "Hz"
. . . . "Transition Band:" = FILT(3,ITYPE) "Hz"
. . . . "Stopband Attenuation:" = FILT(4,ITYPE) "dB"
. . . Else If FILT(1,ITYPE) = 'M":
. . . . "Median Filter"
. . . . "Median Window Length:" = FILT(2,ITYPE) "samples"
. . . End if
. . End if
. [End caption]
. If PDATE is TRUE Then label plot with current date
    (dd-mon-year);

. Return to calling routine;
```

SETA/ADS Software Development
Power Spectral Density Program

Definitions and Notes

a. Data streams and processing specification parameters are indexed in the following manner:
SETA X-acceleration = 1
SETA Y-acceleration = 2
SETA Z-acceleration = 3
SETA temperature = 4

b. Standard lower limit for plotted frequency range will be zero.

c. Standard lower limit for PSD amplitudes will be zero for linear plot and -6 for logarithmic plot.

d. ISTART = initial index for loading data into buffer;

e. CEILING is a standard Fortran-90 function for the least integer greater than its argument. A defining algorithm is:

```
Procedure CEILING(X):
CEILING = INT(X)
If X > CEILING Then CEILING = CEILING + 1;
End
```

f. NMISS = number of missing points, for removable gap;

g. WTSQ = sum of squares of windowing weight factors;

h. A sample date in this format would be 15-Jul-1993.

SETA/ADS Software Development
Power Spectral Density Program

APPENDIX D - Filtering Program

SETA/ADS Software Development
Filtering Program

Files:
ACCEL - SETA raw accelerometer data in PL format
FILT - SETA filtered accelerometer data in PL format

Parameters:
MAXSMP = 600 [maximum number of samples in a data block (ACCEL or FILT)]
MXMISS = 600 [maximum number of missing samples allowed for removable gap, equivalent to TGAP*SAMPRT]
MXWNDW = 100 [maximum number of samples to be used for median window length]
KFLIM = 1000 [maximum index for filter weights; typical estimate would be (SAMPRT/(PBAND+TBAND))]
LENBUF = 4302 [length of buffer (array) for storage of raw data samples to be filtered; minimum assignment should be MAXSMP + 3*KFLIM + MXMISS + MXWNDW + 2]

Overview:
1. Acquire data values from ACCEL into interim storage (RAWDATA, FLAGVAL), with checking for time gaps;
2. Transfer data to processing buffers (BUF, FLAGS), for "wild point" checking (if enabled) and filtering, with DC extension for gap initialization and termination, or interpolation across removable gaps;
3. Filter acceleration and temperature values, storing filtered data in buffer (FBUF);
4. Transfer filtered data to output buffer, writing data to FILT when complete output blocks are accumulated or time gap is encountered.

{INIT} [Initialization]
. Open ACCEL data file for input;
. Open SPEC parameter file for input;
. Open FILT data file for output;
. Initialize variables:
. . TLAST = 0 [composite day number/time for last sample in filtering buffer]
. . TSTART = 0 [composite day number/time for ISTART sample in filtering buffer]
. . TFIRST = 0 [composite day number/time for IFIRST sample in filtering buffer (first sample to be transferred to interim output buffer)]
. . STAT = blank (' ') [status for data acquisition]

{READ_SPEC} [Read processing specifications from user][a]
. Read user specifications for each data sequence (accelerations and temperature), with defaults in angle brackets:
. . FTYPE(4) <'L','L','L','M'> [Filter types (Low-pass or Median)]

```
. . PBAND(4) <0.05, 0.05, 0.10, 0.00> [Passband width, in Hz
      (low-pass filter only)]
. . TBAND(4) <0.05, 0.05, 0.10, 0.00> [Transition band width, in
      Hz (low-pass filter only)]
. . ATTEN(4) <45.0, 45.0, 45.0, 1.00> [Stopband attenuation, in
      decibels (low-pass filter only)]
. . MWNDW(4) <20, 20, 20, 20> [Median window length, in samples,
      for median filtering or "wild point" removal]
. . WPTHR(4) <40000.0, 40000.0, 40000.0, 40000.0> [Threshold
      level, in micro-Gees or degrees Celsius, for "wild point"
      removal]
. . WPEDIT(4) <'Y','Y','Y','Y'> [Flags for "wild point" editing]
. . TGAP <1.00> [Gap threshold, in seconds]
Notes:
  1)  It would also be possible to specify a minimal data segment
      length, in seconds, but this option will be excluded unless
      justified by many data sequences.
  2)  "Wild point" editing in conjunction with median filtering
      is redundant.

{READ_HDR} [Read and store header information for input file]
. Read Raw Data header items from file ACCEL: [See data format
    descriptions]
. . EXPID
. . DTYPE
. . SAMPRT
. . DECIM
. . ASCALE
. . BEGYR
. . BEGMON
. . BEGDAY
. . BEGDN
. . GENDN
. . Blank words
. [Initialize header-dependent variables]
. TINT = 1.0/SAMPRT [time interval between samples, in seconds]
. TINC = TINT/86400.0 [composite day number/time for sampling
    interval]

{BLD_FILT} [Calculate the filter weights and sample count
  duration, based on specified design parameters (one set for
  each of X-acceleration, Y-acceleration, Z-acceleration, and
  temperature)]
. For I = 1 to 4
. . If FTYPE(I) = 'L' Then
. . . [Define a low-pass Kaiser filter]
. . . Invoke KAISER(PBAND(I)/SAMPRT, TBAND(I)/SAMPRT, ATTEN(I),
        KF(I), FW(0,I), KFLIM) with input parameters PBAND(I),
        TBAND(I), ATTEN(I), and KFLIM to determine the filter
        length LF(I) in samples and the (right-half) filter
```

```
          weights (FW(J,I), J = 0, KF(I)), with LF(I) = 2*KF(I) +
          1, and KF(I) ≤ KFLIM;(b)
. . . If LF(I)+1 > LENBUF Then
. . . . Print error report for user, including input
          specifications for this filter;
. . . . Terminate program with error status;
. . . End if
. . Else If FTYPE(I) = 'M' Then
. . . LF(I) = MWNDW(I)
. . . KF(I) = 0
. . . If LF(I) + 1 > LENBUF Then
. . . . Print error report for user, including input
          specifications for this filter;
. . . . Terminate program with error status;
. . . End if
. . Else
. . . Report error in specification of FTYPE(I);
. . . Terminate program with error status;
. . End if
. Next I

{START} [Set data/specification-dependent starting values for
  processing]
. [Determine the largest filter length, for manipulating data]
. MAXKF = Max({KF(I), I = 1 to 4})
. [Define initial index for loading data into BUF]
. MAXKF1 = MAXKF + 1 [utility variable]
. ISTART = MAXKF1 [initial index at which to start storing data]
. IEND = ISTART [index at which data storage ends (for end of BUF
    or time gap, including end-of-data)]
. TNEXT = BEGDN - 1 [composite day number/time for next sample
    expected for filtering buffer; initialized here to avoid
    overwhelmingly large values]

{WRITE_HDR} [Write header for output filtered data file]
. Calculate FILTDN as SETA day number for date of filter
    processing;
. Write Filtered Data header items to file FILT: [See data format
    descriptions]
. . EXPID
. . DTYPE
. . SAMPRT
. . DECIM
. . ASCALE
. . BEGYR
. . BEGMON
. . BEGDAY
. . BEGDN
. . GENDN
. . FILTDN
```

```
. . TGAP
. . WPTHRX (= WPTHR(1))
. . WPTHRY (= WPTHR(2))
. . WPTHRZ (= WPTHR(3))
. . WPTHRT (= WPTHR(4))
. . WPEDX (= WPEDIT(1))
. . WPEDY (= WPEDIT(2))
. . WPEDZ (= WPEDIT(3))
. . WPEDT (= WPEDIT(4))
. . FILTX(1) (= FTYPE(1))
. . FILTX(2) (= PBAND(1) if FTYPE(1) = 'L'; = MWNDW(1) if
      FTYPE(1) = 'M')
. . FILTX(3) (= TBAND(1) if FTYPE(1) = 'L'; = "blank" if FTYPE(1)
      = 'M')
. . FILTX(4) (= ATTEN(1) if FTYPE(1) = 'L'; = "blank" if FTYPE(1)
      = 'M')
. . FILTY(1) (= FTYPE(2))
. . FILTY(2) (= PBAND(2) if FTYPE(2) = 'L'; = MWNDW(2) if
      FTYPE(2) = 'M'))
. . FILTY(3) (= TBAND(2) if FTYPE(2) = 'L'; = "blank" if FTYPE(2)
      = 'M')
. . FILTY(4) (= ATTEN(2) if FTYPE(2) = 'L'; = "blank" if FTYPE(2)
      = 'M')
. . FILTZ(1) (= FTYPE(3))
. . FILTZ(2) (= PBAND(3) if FTYPE(3) = 'L'; = MWNDW(3) if
      FTYPE(3) = 'M'))
. . FILTZ(3) (= TBAND(3) if FTYPE(3) = 'L'; = "blank" if FTYPE(3)
      = 'M')
. . FILTZ(4) (= ATTEN(3) if FTYPE(3) = 'L'; = "blank" if FTYPE(3)
      = 'M')
. . FILTT(1) (= FTYPE(4))
. . FILTT(2) (= PBAND(4) if FTYPE(4) = 'L'; = MWNDW(4) if
      FTYPE(4) = 'M')
. . FILTT(3) (= TBAND(4) if FTYPE(4) = 'L'; = "blank" if FTYPE(4)
      = 'M')
. . FILTT(4) (= ATTEN(4) if FTYPE(4) = 'L'; = "blank" if FTYPE(4)
      = 'M')

{FILL_BUF} [Fill the buffer with new data samples]
. Invoke LOAD_BUF(BUF, FLAGS, LENBUF, ASCALE, TSTART, TLAST,
    TNEXT, TINC, TGAP, ISTART, MAXKF, IEND, MBEG, MEND, STAT) to
    fill buffers for each data sequence (or at least fill to time
    gap);
. If STAT = 'DONE' Then proceed to {END_FILT};

{REPL_WILD} [Discover "wild points" in acquired segment, and
  replace by local median]
. For I = 1 to 4
. . If WPEDIT(I) = 'Y' Then
. . . If STAT = 'FILL' Then
```

```
. . . . [Checking for "wild points" before gap is redundant]
. . . . Invoke WP_CHECK(BUF, I, FLAGS, MEND+1, IEND, MWNDW(I),
          WPTHR(I)) to edit "wild points" for selected data
          sequence, following removable gap;
. . . Else
. . . . Invoke WP_CHECK(BUF, I, FLAGS, ISTART, IEND, MDWND(I),
          WPTHR(I)) to edit "wild points" for selected data
          sequence;
. . . End if
. . End if
. Next I

{PAD_BUF} [Determine the appropriate extension or interpolation
  requirements for each buffer, and perform the corresponding
  padding]
. ISTOP = IEND - MAXKF [final index for filtering or from which
    to store data into OUTDATA]
. If STAT = 'INIT' Then
. . {INIT_FILT} [Perform DC extension at the beginning of the
      data sequence, to avoid losing the initial data points by
      filtering]
. . For I = 1 to 4
. . . For J = 1 to ISTART-1 [should have ISTART-1 = MAXKF, from
        initialization or STORE_BUF]
. . . . BUF(J,I) = BUF(ISTART,I)
. . . Next J
. . Next I
. Else if STAT = 'TERM' Then
. . {TERM_FILT} [Perform DC extension at the end of the data
      sequence, to avoid losing the trailing data points by
      filtering]
. . For I = 1 to 4
. . . For J = ISTART to IEND+MAXKF
. . . . BUF(J,I) = BUF(IEND,I)
. . . Next J
. . Next I
. . [Special case for setting last index of data to be stored]
. . ISTOP = IEND
. Else if STAT = 'FILL' Then
. . {INTERP} [Interpolate between two samples on either side of a
      removable gap]
. . For I = 1 to 4
. . . DSTEP = (BUF(MEND+1,I) - BUF(MBEG-1,I))/(MEND - MBEG + 2)
. . . For J = MBEG to MEND
. . . . BUF(J,I) = BUF(J-1,I) + DSTEP
. . . Next J
. . Next I
. . For J = MBEG to MEND
. . . FLAGS(J) = F000h [set flag values for interpolation]
. . Next J
```

```
. End if

{FILT_SAMP} [Generate output samples for each acquired sample, up
  to the semi-duration of the longest filter, and write the
  filtered samples to the output file]
. For I = 1 to 4
. . If FTYPE(I) = 'L' Then
. . . For J = MAXKF1 to ISTOP
. . . . FBUF(J,I) = FW(0,I)*BUF(J,I)
. . . . For K = 1 to KF(I)
. . . . . FBUF(J,I) = FBUF(J,I) + FW(K,I)*(BUF(J-K,I) +
            BUF(J+K,I))
. . . . Next K
. . . Next J
. . Else if FTYPE(I) = 'M' Then
. . . Invoke MED_FILT(BUF, I, MAXKF1, ISTOP, MWNDW(I), FBUF) to
        perform median filtering for sequence, in disjoint groups
        of MWNDW(I) samples;
. . End if
. Next I
. If STAT = 'FILL' Then
. . TFIRST = TSTART - (MEND - MAXKF)*TINC
. Else
. . TFIRST = TSTART - (ISTART - MAXKF1)*TINC
. End if
. If ISTOP > MAXKF1 Then
. . Invoke STORE_BUF(FBUF, FLAGS, BUF, LENBUF, ASCALE, TFIRST,
      MAXKF1, IEND, ISTART, ISTOP, STAT, MAXKF, TINT) to store
      the filtered data in FILT;
. Else
. . [Retain data, but advance starting index]
. . ISTART = IEND + 1
. End if
. Proceed from {FILL_BUF};

{END_FILT} [Conclude processing]
. Close output file FILT;
. Report program conclusion;
```

Subroutines

LOAD_BUF(BUF, FLAGS, LENBUF, ASCALE, TSTART, TLAST, TNEXT, TINC, TGAP, ISTART, MAXKF, IEND, MBEG, MEND, STAT)
BUF(LENBUF,4) = (R*4) array for individual data sequences (X,Y,Z,T) [output];
FLAGS(LENBUF) = (I*2) array for range and interpolation flags [output];
LENBUF = (I*2) time-sequence dimension for BUF (maximum number of samples) [input];
ASCALE = (R*4) scale factor for stored acceleration counts to micro-Gees [input];
TSTART = (R*8) composite day/time for beginning of current data group acquired from ACCEL [output];
TLAST = (R*8) composite day number/time for last sample in filtering buffer [input/output];
TNEXT = (R*8) composite day number/time for next sample expected for filtering buffer [input/output];
TINC = (R*8) composite day number/time for sampling interval [input]
TGAP = (R*4) gap threshold, in seconds [input]
ISTART = (I*2) initial index at which to start storing data [input];
MAXKF = (I*2) maximum filter weight dimension, for indexing to padded position in buffer [input];
IEND = (I*2) index at which data storage ends (for end of BUF or time gap, including end-of-data) [output];
MBEG = (I*2) initial index at which replaceable missing data occurs [output];
MEND = (I*2) last index at which replaceable missing data occurs [output];
STAT = (C*4) status of data acquisition [output]:
'INIT' = data acquired after time gap;
'OKAY' = data acquired with no immediately preceding time gap;
'TERM' = data ends at time gap;
'FILL' = data segment contains removable time gap;
'DONE' = all input values have been acquired;

Parameters:
MAXSMP = 600 [maximum number of samples in a data block (ACCEL)]

Local variables:
DATDN = (I*2) SETA day number for beginning of data block
DATTIM = (I*4) time of day for beginning of data block, in tenths of seconds
RAWDATA(MAXSMP,4) = (I*2) scaled accelerations and temperature
FLAGVAL(MAXSMP) = (I*2) packed range flags
NMISS = (I*4) calculated number of missing samples, based on gap in time sequence
MENDI4 = (I*4) projected last index at which missing data

```
    occurs (removable or permanent gap)
Initialization values:
  IOSTAT = 0 [file read return status]
  NREM = 0 [number of samples remaining for transfer to BUF]
  NSAMP = 0 [number of samples in data group acquired from ACCEL]
  KREM = 1 [index of first sample remaining after incomplete
    transfer to BUF]

. If IOSTAT = -1 Then
. . Set STAT = 'DONE'
. . Return to calling routine;
. End if
. {FETCH} Set IOSTAT = 0 [initialize for successful read];
. If NREM = 0 Then
. . Read DATDN, DATTIM, NSAMP, ((RAWDATA(K,L), L = 1, 4),
      FLAGVAL(K), K = 1, NSAMP) from ACCEL;
. . If end-of-file on read Then
. . . [This should not happen during a data block without error];
. . . Set IOSTAT = -1; [standard FORTRAN result]
. . . Set STAT = 'TERM';
. . . IEND = ISTART - 1
. . . [Special case for assignment of time at ISTART]
. . . TSTART = TNEXT
. . . Return to calling routine;
. . Else if error on read Then
. . . Set IOSTAT = error number;
. . . Report error in data acquisition;
. . . Terminate program with error status;
. . End if
. . CURRDT = DATDN + DATTIM/864000.0
. End if
. [Compare current initial day/time for block to expected
    day/time]
. If |CURRDT - TNEXT| > 0.5*TINC Then
. . [A time gap exists (or the data sequence has just begun)]
. . MBEG = ISTART
. . NMISS = Round((CURRDT - TNEXT)/TINC) [must be at least one,
      by original test condition; can be very large]
. . MENDI4 = ISTART + NMISS - 1
. . If |CURRDT - TNEXT| > TGAP/86400.0 or MENDI4 ≥ LENBUF Then
. . . [This is a permanent gap]
. . . If TLAST = 0 Then
. . . . Set STAT = 'INIT';
. . . . IEND = Min(ISTART+NSAMP-1, LENBUF)
. . . . NREM = NSAMP - (IEND - ISTART) - 1
. . . . [Load processing buffer]
. . . . For I = 1 to 3
. . . . . K = KREM
. . . . . For J = ISTART to IEND
. . . . . . BUF(J,I) = ASCALE*RAWDATA(K,I)
```

```
. . . . . . . K = K + 1
. . . . . . Next J
. . . . . Next I
. . . . . K = KREM
. . . . . For J = ISTART to IEND
. . . . . . BUF(J,4) = RAWDATA(K,4)
. . . . . . FLAGS(J) = FLAGVAL(K)
. . . . . . K = K + 1
. . . . . Next J
. . . . . MBEG = IEND
. . . . . MEND = IEND
. . . . Else
. . . . . Set STAT = 'TERM';
. . . . . IEND = ISTART - 1
. . . . . [Special case for assignment of time at ISTART]
. . . . . TSTART = TNEXT
. . . . . NREM = NSAMP
. . . . . TLAST = 0 [set to trigger storage re-initialization]
. . . . . MBEG = ISTART
. . . . . MEND = ISTART
. . . . . Return to calling routine;
. . . . End if
. . . Else [This is a removable time gap]
. . . . Set STAT = 'FILL';
. . . . [Insure that removable gap does not straddle upper index
          limit of buffer, thus impeding interpolations]
. . . . MEND = MENDI4 [limit TGAP so that MEND < 32768]
. . . . IEND = Min(MEND+NSAMP, LENBUF)
. . . . NREM = NSAMP - (IEND - MEND)
. . . . [Load processing buffer]
. . . . For I = 1 to 3
. . . . . K = KREM
. . . . . For J = MEND+1 to IEND
. . . . . . BUF(J,I) = ASCALE*RAWDATA(K,I)
. . . . . . K = K + 1
. . . . . Next J
. . . . Next I
. . . . K = KREM
. . . . For J = MEND+1 to IEND
. . . . . BUF(J,4) = RAWDATA(K,4)
. . . . . FLAGS(J) = FLAGVAL(K)
. . . . . K = K + 1
. . . . Next J
. . . End if
. . Else
. . . [No time gap]
. . . Set STAT = 'OKAY';
. . . IEND = Min(ISTART+NSAMP-1, LENBUF)
. . . NREM = NSAMP - (IEND - ISTART) - 1
. . . [Load processing buffer]
```

```
. . For I = 1 to 3
. . . K = KREM
. . . For J = ISTART to IEND
. . . . BUF(J,I) = ASCALE*RAWDATA(K,I)
. . . . K = K + 1
. . . Next J
. . Next I
. . K = KREM
. . For J = ISTART to IEND
. . . BUF(J,4) = RAWDATA(K,4)
. . . FLAGS(J) = FLAGVAL(K)
. . . K = K + 1
. . Next J
. . MBEG = IEND
. . MEND = IEND
. End if
. TSTART = CURRDT
. TLAST = CURRDT + (NSAMP - NREM - 1)*TINC [time of last sample
    in BUF]
. TNEXT = TLAST + TINC
. CURRDT = TNEXT [update current time for remaining samples]
. KREM = Mod(NSAMP - NREM + 1, NSAMP) [first index of remaining
    samples]
. NSAMP = NREM
. Return to calling routine;
```

```
WP_CHECK(BUF, I, FLAGS, ISTART, IEND, MWNDW, WPTHR)
  BUF = array for individual data sequence [input/output];
  I = selection index for data type (acceleration or temperature)
  FLAGS = array for range and interpolation flags [input/output];
  ISTART = index of first sample to be checked;
  IEND = index of last sample to be checked;
  MWNDW = number of samples for median referencing;
  WPTHR = threshold level for wild point exclusion;
  [see existing EDITDTA routine, but use comparison to WPTHR
    rather than local standard deviations]
  [also check condition MWNDW > IEND - ISTART + 1]
  [set flag bit for edited samples]
```

```
STORE_BUF(FBUF, FLAGS, BUF, LENBUF, ASCALE, TFIRST, IFIRST, IEND,
  ISTART, ISTOP, STAT, MAXKF, TINT)
  FBUF(LENBUF,4) = (R*4) array for individual filtered data
    sequences (X,Y,Z,T) [input/output];
  FLAGS(LENBUF) = (I*2) array for range and interpolation flags
    [input/output];
  BUF(LENBUF,4) = (R*4) array for individual raw data sequences
    (X,Y,Z,T) [input/output];
  LENBUF = (I*2) time-sequence dimension for BUF and FBUF
```

```
    (maximum number of samples) [input];
  ASCALE = (R*4) scale factor for stored acceleration counts to
    micro-Gees [input];
  TFIRST = (R*8) composite day/time for beginning of current
    filtered data segment in FBUF [input];
  IFIRST = (I*2) initial index from which to start storing data
    into OUTDATA [input];
  IEND = (I*2) index at which data storage ends (for end of
    BUF/FBUF or time gap, including end-of-data) [input];
  ISTART = (I*2) initial index at which to start loading data
    into BUF [input/output];
  ISTOP = (I*2) final index from which to store data into OUTDATA
    [input/output];
  STAT = (C*4) status of data acquisition [input]:
    'INIT' = data acquired after time gap;
    'OKAY' = data acquired with no immediately preceding time
      gap;
    'TERM' = data ends at time gap;
    'FILL' = data segment contains removable time gap;
    'DONE' = all input values have been acquired;
  MAXKF = (I*2) maximum filter weight dimension, for indexing to
    padded position in buffer [input];
  TINT = (R*4) time interval between samples, in seconds;
  MAXSMP = (I*2) maximum number of samples in a data block;
Parameters:
  MAXSMP = 600 [maximum number of samples in a data block (FILT)]
Local variables:
  OUTDN = (I*2) SETA day number for beginning of data block
  OUTTIM = (I*4) time of day for beginning of data block, in
    tenths of seconds
  OUTDATA(MAXSMP,4) = (I*2) scaled accelerations and temperature
  FLAGOUT(MAXSMP) = (I*2) packed range flags
  IMOVE = (I*2) initial index for data samples to be shifted
Initialization values:
  NOUT = 0  [number of samples in OUTDATA available for output]
  TCHECK = 0 [projected time for IFIRST sample]

. [Load storage buffer, with output to FILT when full]
. If NOUT = 0 Then
. . [Define the date and time for the current output group]
. . OUTDN = Integer(TFIRST)
. . OUTTIM = Round(864000*Fraction(TFIRST)) [for time in tenths
      of seconds]
. Else
. . [Check the initial time for the newly acquired group against
      its expected initial time, as predicted from the current
      output buffer]
. . If |TFIRST - TCHECK| > 0.5*TINT/86400.0 Then
. . . [Report the discrepancy (for now); may need to revise
        output blocking process if time tag slippage occurs]
```

```
. . . Report TFIRST, TCHECK, and discrepancy;
. End if
. K = NOUT + 1
. For J = IFIRST to ISTOP
. . For I = 1 to 3
. . . OUTDATA(K,I) = Round(FBUF(J,I)/ASCALE) [accelerations]
. . Next I
. . OUTDATA(K,4) = Round(FBUF(J,4)) [temperatures]
. . FLAGOUT(K) = FLAGS(J) [range and edit flags]
. . K = K + 1
. . If K > MAXSMP Then
. . . NOUT = K - 1
. . . Write OUTDN, OUTTIM, NOUT, ((OUTDATA(M,L), L = 1, 4),
        FLAGOUT(M), M = 1, NOUT) to FILT;
. . . [Time tagging is mostly performed here, for continuous
        output sequence]
. . . OUTTIM = OUTTIM + Round(10*NOUT*TINT) [time of day in
        tenths of seconds]
. . . OUTDN = OUTDN + Integer(OUTTIM/864000.0)
. . . OUTTIM = Mod(OUTTIM,864000.0)
. . . [Reset variables for next output group]
. . . NOUT = 0
. . . K = 1
. . End if
. Next J
. NOUT = K - 1
. If STAT = 'TERM' and NOUT > 0 Then
. . [Write the current output buffer to FILT, even if only
      partially full]
. . Write OUTDN, OUTTIM, NOUT, ((OUTDATA(M,L), L = 1, 4),
      FLAGOUT(M), M = 1, NOUT) to FILT;
. . NOUT = 0
. End if
. [Calculate the next expected sample day/time from FBUF, for
    checking (not valid for STAT = 'TERM', when next time will be
    re-initialized by NOUT = 0)]
. TCHECK = OUTDN + Double(OUTTIM/864000.0) +
    Double(NOUT*TINT/86400.0) [insure double precision for
    intermediate results](c)

{SHIFT} ["Shift" the filtered samples out of the current buffer,
  by redefining indices, with the padding and unfiltered samples
  moving to the beginning of the buffer]
. If STAT = 'TERM' Then
. . [The remaining values are just DC extension, and the leading
      pad values will be determined later, so re-initialize for
      resumption after gap]
. . ISTART = MAXKF + 1
. Else
. . IMOVE = ISTOP + 1 - MAXKF
```

```
. . For I = 1 to 4
. . . K = 1
. . . For J = IMOVE to IEND
. . . . BUF(K,I) = BUF(J,I)
. . . . FBUF(K,I) = FBUF(J,I)
. . . . K = K + 1
. . . Next J
. . Next I
. . K = 1
. . For J = IMOVE to IEND
. . . FLAGS(K) = FLAGS(J)
. . . K = K + 1
. . Next J
. . ISTART = K
. End if
. Return to calling routine;
```

```
KAISER(PBAND, TBAND, ATTEN, KF, FW, KFLIM)
[Define filtering weights for a Kaiser filter, based on supplied
  parameters, with PBAND and TBAND in normalized frequency units
  (based on sampling rate)]
. [See DNER2]
```

```
MED_FILT(BUF, I, ISTART, ISTOP, MWNDW, FBUF)
[Perform median filtering on a designated data sequence using a
  specified subset size]
. [See EDITDTA]
```

SETA/ADS Software Development
Filtering Program

Definitions and Notes

a. Data streams and processing specification parameters are indexed in the following manner:
SETA X-acceleration = 1
SETA Y-acceleration = 2
SETA Z-acceleration = 3
SETA temperature = 4

b. FW(0:KFLIM,4) = up to KFLIM + 1 filter weights for time-symmetric low-pass filters, for accelerations and temperature
KF(4) = actual index limit for each of the four sets of filter weights
LF(4) = actual filter duration limits, in terms of number of samples, for each of the four sets of filter weights

c. "Double" function will return a double precision value of its argument

SETA/ADS Software Development
Filtering Program

APPENDIX E - Ephemeris and Attitude Merge Program

Files:
  FILT - SETA filtered accelerometer data in PL format
  EPHEM - Ephemeris Agency Data File
  EVENT - Event Agency Data File
  HEADER - Header Agency Data File
  MERGE - SETA merged data in PL format
  LOG - log file of gap endpoints and processing status
  THRUST - thruster and torqrod listing file

Parameters:
  MAXINP = 600 [maximum number of samples in an input data block (FILT)]
  MAXOUT = 64 [maximum number of samples in an output data block (MERGE)]

Overview:
1. Acquire data values from FILT into interim storage (FILTVAL, FLAGVAL), with checking for time gaps;
2. Determine time and index for data samples to be transferred to output, at requested decimation;
3. Determine bracketing ephemeris, event, and header samples, and assign required quantities to time of data sample (by interpolation or nearest occurrence), with appropriate transformations;
4. Transfer merged data to output file, writing data to MERGE when complete output blocks are accumulated or time gap is encountered.

{OPTS} [Obtain user specifications]
. Read processing options from file or terminal, or retain defaults, in parentheses:
. . MDECIM (= 1) [decimation factor to be applied to filtered data (to be selected consistent with filter duration)]

{INIT} [Initialization]
. Open FILT data file for input;
. If error on open Then terminate program, with error message;
. Open EPHEM parameter file for input;
. If error on open Then terminate program, with error message;
. Open EVENT parameter file for input;
. If error on open Then terminate program, with error message;
. Open HEADER parameter file for input;
. If error on open Then terminate program, with error message;
. Open MERGE data file for output;
. Open LOG listing file for output;
. Open THRUST listing file for output;
. Initialize variables, with data acquisition:
. . {INIT_EPH} Read (EPH(I), I = 1 to 4) from EPHEM [see EPH structure definition on page 109];

```
. . For I = 1 to 4
. . . EPHT(I) = NDAYS(IYY(I),IMM(I),IDD(I)) + (3600*IHH(I) +
        60*IMN(I) + ISS(I))/86400.0(a)
. . Next I

. . {INIT_EVT}
. . For I = 1 to 4
. . . Invoke GETEVT(EVT(I),EVTSTAT) to acquire initial data
        records from EVENT, for ACS data records only, skipping
        CDH and HSC data [see structure definition in Software
        Development document];
. . . If EVTSTAT ≠ 0 Then
. . . . Report EVTSTAT to user and LOG, as EVENT error status;
. . . . If I > 1 Then report EVTT(I-1) to user and LOG, as last
          time successfully acquired;
. . . . Proceed from {END_MERGE};
. . . End If
. . . EVTT(I) = NDAYS(IVYY(I),IVMM(I),IVDD(I)) + (3600*IVHH(I) +
        60*IVMN(I) + IVSS(I))/86400.0(b)
. . Next I

. . {INIT_HDR} [Acquire the relevant subset of the HEADER data,
      in HDR]
. . For I = 1 to 4
. . . Invoke GETHDR(HDR(I),HDRSTAT) to acquire initial data
        records from HEADER [see structure definition in Software
        Development document];
. . . If HDRSTAT ≠ 0 Then
. . . . Report HDRSTAT to user and LOG, as HEADER error status;
. . . . If I > 1 Then report HDRT(I-1) to user and LOG, as last
          time successfully acquired;
. . . . Proceed from {END_MERGE};
. . . End If
. . . HDRT(I) = NDAYS(IHYY(I),IHMM(I),IHDD(I)) + (3600*IHHH(I) +
        60*IHMN(I) + IHSS(I))/86400.0(c)
. . Next I

. . NOUT = 0(d)
. . DTPREV = 0(e)
. . DTLAST = 0(f)

{READ_HDR} [Read and store header information for input file]
. Read Filtered Data header items from file FILT: [See data
    format descriptions]
. . EXPID
. . DTYPE
. . SAMPRT
. . DECIM
. . ASCALE
. . BEGYR
```

```
. . BEGMON
. . BEGDAY
. . BEGDN
. . GENDN
. . FILTDN
. . TGAP
. . WPTHRX
. . WPTHRY
. . WPTHRZ
. . WPTHRT
. . WPEDX
. . WPEDY
. . WPEDZ
. . WPEDT
. . FILTX(K), K = 1 to 4
. . FILTY(K), K = 1 to 4
. . FILTZ(K), K = 1 to 4
. . FILTT(K), K = 1 to 4
. [Report date of data]
. Write 'Data Source', DTYPE, BEGYR, BEGMON, BEGDAY to LOG and
    THRUST
. [Initialize header-dependent variables]
. TINT = DECIM/SAMPRT [time interval between samples, in seconds]
. TINC = TINT/86400.0 [composite day number/time for sampling
    interval]
. [Determine cumulative decimation factor]
. DECIM2 = MDECIM*DECIM;

{WRITE_HDR} [Write header for output merged data file, identical
  in form to density header, but with blank words as place-
  holders]
. Obtain processing date (current date) from system, as IPYR
    [year], IPMON [month], IPDAY [day of month];
. MRGDN = NDAYS(IPYR,IPMON,IPDAY) [as character string]
. DTYPE = 'MERGE   '
. DENDN = '     '
. AREF = '         '
. POS1 = (TBD)
. POS2 = (TBD)
. POS3 = (TBD)
. CALOPT = '  '
. Write Merged Data header items to file MERGE: [See data format
    descriptions]
. . EXPID
. . DTYPE
. . SAMPRT
. . DECIM2
. . ASCALE
. . BEGYR
. . BEGMON
```

```
. . BEGDAY
. . BEGDN
. . GENDN
. . FILTDN
. . MRGDN
. . DENDN
. . TGAP
. . WPTHRX
. . WPTHRY
. . WPTHRZ
. . WPTHRT
. . WPEDX
. . WPEDY
. . WPEDZ
. . WPEDT
. . FILTX(K), K = 1 to 4
. . FILTY(K), K = 1 to 4
. . FILTZ(K), K = 1 to 4
. . FILTT(K), K = 1 to 4
. . AREF
. . POS1
. . POS2
. . POS3
. . CALOPT

{GET_DATA} [Acquire filtered accelerometer data block]
. Read DATDN, DATTIM, NSAMP, ((FILDATA(K,L), K = 1, 4), L = 1,
    NSAMP) from FILT [See data format descriptions];
. If end-of-file on read Then
. . [This should not happen during a data block without error];
. . Report end-of-file to user;
. . If NOUT > 0 Then write MRGDN(NOUT), MRGTIM(NOUT)/10,
      ORBNUM(NOUT), ALT(NOUT), LEG(VRAD(NOUT)) to LOG;
. . Proceed to {END_MERGE};
. Else If error on read Then
. . Report error (with error number) to user;
. . If NOUT > 0 Then write MRGDN(NOUT), MRGTIM(NOUT)/10,
      ORBNUM(NOUT), ALT(NOUT), LEG(VRAD(NOUT)) to LOG;
. . Report error (with error number) to LOG;
. . Proceed to {END_MERGE};
. End If

{GAP_CHK} [Check for time gap from previous accelerometer data
  block]
. DTFIRST = DATDN + DATTIM/86400.0(g)
. If (DTFIRST - DTPREV) > DECIM*TINC Then
. . [A time gap exists between data blocks, so reinitialize the
      decimated time track, if necessary, and generate the
      appropriate reports]
. . If (DTFIRST - DTLAST) > DECIM2*TINC Then
```

```
. . . [This time gap will propagate to the MERGE data]
. . . ISEL = 1(h)
. . . RPTBEG = TRUE(i)
. . . [Report the last sample of the previous MERGE block, and
        write the data block to MERGE]
. . . If NOUT > 0 Then
. . . . Write MRGDN(NOUT), MRGTIM(NOUT)/10, ORBNUM(NOUT),
          ALT(NOUT), LEG(VRAD(NOUT)) to LOG;
. . . . Write NOUT, (MRGDATA(L), L = 1, NOUT) to MERGE [see
          MRGDATA structure definition on page 112];
. . . . NOUT = 0
. . . End If
. . Else
. . . [The gap will not be evident on the decimated time track,
        but the selection index from the FILT data must be reset]
. . . DTNEXT = DTLAST + DECIM2*TINC(j)
. . . ISEL = Round((DTNEXT - DTFIRST)/(DECIM*TINC) + 1)
. . End If
. End If
. [Update day/time for last FILT sample]
. DTPREV = DTFIRST + (NSAMP - 1)*TINC

{MERGE_LOOP} [Merge selected samples of filtered data with
  ephemeris, header, and event data]
. For I = ISEL to NSAMP, by MDECIM
. . DTREF = DTFIRST + (I - 1)*TINC(k)

. . {EPH_BRACKET} [For selected input sample, find the bracketing
        ephemeris records, for interpolation or nearest item]
. . If DTREF < EPHT(2) Then
. . . [There is a problem with the EPHEM coverage, which starts
        within a minute after the accelerometer data or has a
        gap, or a time reversal has occurred in the FILT data]
. . . Report DTREF, EPHT to user and LOG, with error message
        about EPHEM coverage;
. . . Proceed to {END_MERGE};
. . Else If DTREF ≥ EPHT(3) Then
. . . {GET_EPH} [Read EPHEM until bracketing times are acquired]
. . . [Shuffle reference samples to prepare for new acquisition]
. . . For J = 1 to 3
. . . . EPH(J) = EPH(J+1)
. . . . EPHT(J) = EPHT(J+1)
. . . Next J
. . . Read EPH(4) from EPHEM;
. . . If end-of-file or error on read Then
. . . . Report error to user and LOG;
. . . . Report EPHT(3) to user and LOG, as last time successfully
          acquired;
. . . . Proceed to {END_MERGE};
. . . End If
```

```
. . . EPHT(4) = NDAYS(IYY(4),IMM(4),IDD(4)) + (3600*IHH(4) +
        60*IMN(4) + ISS(4))/86400.0
. . . If DTREF ≥ EPHT(3) Then proceed from {GET_EPH};
. . End If

. . {EVT_BRACKET} [For selected input sample, find the bracketing
      event records, for interpolation or nearest item]
. . If DTREF < EVTT(2) Then
. . . [There is a problem with the EVT coverage, which starts
        within a minute after the accelerometer data or has a
        gap, or a time reversal has occurred in the FILT data]
. . . Report DTREF, EVTT to user and LOG, with error message
        about EVT coverage;
. . . Proceed to {END_MERGE};
. . Else If DTREF ≥ EVTT(3) Then
. . . {GET_EVT} [Read EVT until bracketing times are acquired]
. . . [Shuffle reference samples to prepare for new acquisition]
. . . For J = 1 to 3
. . . . EVT(J) = EVT(J+1)
. . . . EVTT(J) = EVTT(J+1)
. . . Next J
. . . Invoke GETEVT(EVT(4),EVTSTAT) to acquire ACS record from
        EVENT, skipping other data types;
. . . If EVTSTAT ≠ 0 Then
. . . . Report EVTSTAT to user and LOG, as EVENT status error;
. . . . Report EVTT(3) to user and LOG, as last time successfully
          acquired;
. . . . Proceed to {END_MERGE};
. . . End If
. . . EVTT(4) = NDAYS(IVYY(4),IVMM(4),IVDD(4)) + (3600*IVHH(4) +
        60*IVMN(4) + IVSS(4))/86400.0
. . . If DTREF ≥ EVTT(3) Then proceed from {GET_EVT};
. . End If

. . {HDR_BRACKET} [For selected input sample, find the bracketing
      header records, for interpolation or nearest item]
. . If DTREF < HDRT(2) Then
. . . [There is a problem with the HEADER coverage, which starts
        within a minute after the accelerometer data or has a
        gap, or a time reversal has occurred in the FILT data]
. . . Report DTREF, HDRT to user and LOG, with error message
        about HEADER coverage;
. . . Proceed to {END_MERGE};
. . Else If DTREF ≥ HDRT(3) Then
. . . {GET_HDR} [Read HEADER until bracketing times are acquired]
. . . [Shuffle reference samples to prepare for new acquisition]
. . . For J = 1 to 3
. . . . HDR(J) = HDR(J+1)
. . . . HDRT(J) = HDRT(J+1)
. . . Next J
```

```
. . . Invoke GETHDR(HDR(4),HDRSTAT) to acquire new data record
        from HEADER;
. . . If HDRSTAT ≠ 0 Then
. . . . Report HDRSTAT to user and LOG, as HEADER error status;
. . . . Report HDRT(3) to user and LOG, as last time successfully
          acquired;
. . . . Proceed to {END_MERGE};
. . . End If
. . . HDRT(4) = NDAYS(IHYY(4),IHMM(4),IHDD(4)) + (3600*IHHH(4) +
        60*IHMN(4) + IHSS(4))/86400.0
. . . If DTREF ≥ HDRT(3) Then proceed from {GET_HDR};
. . End If

. . [A time bracket exists or has been generated for each
      reference file, so interpolate or match data items]
. . NOUT = NOUT + 1

. . {REF_ARRAYS} [Define reference array values from ephemeris,
      event, and header information]
. . For J = 1 to 4
. . . EPHALT(J) = EPH(J).ALT
. . . [Convert latitudes and longitudes to degrees from stored
        units]
. . . EPHLAT(J) = 10^-6×EPH(J).GLAT
. . . EPHLON(J) = 10^-6×EPH(J).GLON
. . . EPHRMAG(J) = EPH(J).RMAG
. . . EPGMLAT(J) = 10^-6×EPH(J).GMLAT
. . . EPGMLON(J) = 10^-6×EPH(J).GMLON
. . . EPGMLT(J) = EPH(J).GMLT
. . . ATTROLL(J) = ATTERR(J).ERR(1) ?[1]
. . . ATTPTCH(J) = ATTERR(J).ERR(2) ?
. . . ATTYAW(J) = ATTERR(J).ERR(3) ?
. . . RATEROLL(J) = ATTEST(J).RATE(1) ?[2]
. . . RATEPTCH(J) = ATTEST(J).RATE(2) ?
. . . RATEYAW(J) = ATTEST(J).RATE(3) ?
. . . For K = 1 to 2
. . . . [Convert the eclipse times to day and fraction]
. . . . DTPENE(J,K) = CVTBT(HDR(J).ENTPEN(K))(l)
. . . . DTUMBE(J,K) = CVTBT(HDR(J).ENTUMB(K))(m)
. . . . DTUMBX(J,K) = CVTBT(HDR(J).LVUMB(K))(n)
. . . . DTPENX(J,K) = CVTBT(HDR(J).LVPEN(K))(o)
. . . Next K
. . Next J
```

---

[1] The attitude angles may be obtained from a parametrized fit routine, to be supplied by DET2/SMC or TRW.

[2] The attitude rates may be obtained from a parametrized fit routine, to be supplied by DET2/SMC or TRW.

```
. . {TAG_DATA} [Provide ephemeris, event, and header information
      for selected sample]
. . MRGDN(NOUT) = Integer(DTREF)
. . MRGTIM(NOUT) = Round(864000*(DTREF - DATDN(NOUT)))
. . ACCX(NOUT) = FILDATA(1,I)
. . ACCY(NOUT) = FILDATA(2,I)
. . ACCZ(NOUT) = FILDATA(3,I)
. . RNGFLG(NOUT) = FILDATA(4,I)
. . ORBNUM(NOUT) = HDR.REV(2)
. . ALT(NOUT) = Round(CINTRP(EPHT,EPHALT,DTREF))

. . {LAT_LON} [Perform two-point great circle interpolation in
      three dimensions to obtain latitude and longitude]
. . . Invoke GCINTS(EPHT(2),EPHLAT(2),EPHLON(2),DTREF,RLAT,RLON)
        to calculate RLAT and RLON at DTREF;
. . . [Convert to output storage units]
. . . LAT(NOUT) = Round(100×RLAT)
. . . LON(NOUT) = Round(100×RLON)

. . RAD(NOUT) = Round(CINTRP(EPHT,EPHRMAG,DTREF))

. . {GLAT_GLON} [Perform two-point great circle interpolation in
      three dimensions to obtain magnetic latitude and longitude]
. . . Invoke GCINTS(EPHT(2),EPGMLAT(2),EPGMLON(2),DTREF,RLAT,
        RLON) to calculate RLAT and RLON at DTREF;
. . . [Convert to output storage units]
. . . GMLAT(NOUT) = Round(100×RLAT)
. . . GMLON(NOUT) = Round(100×RLON)

. . GMLT(NOUT) = Round(0.36*CINTRP(EPHT,EPGMLT,DTREF)) [with
      conversion from micro-hours to tenths of seconds]

. . {SPHERE_VEL} [Calculate the inertial spherical velocity
      components, by interpolation and transformation]
. . . For J = 1 to 3
. . . . For K = 1 to 4
. . . . . EPHXYZ(K) = EPH(K).ECI(J)
. . . . Next K
. . . . ECIPOS(J) = CINTRP(EPHT,EPHXYZ,DTREF)
. . . . For K = 1 to 4
. . . . . EPHXYZ(K) = EPH(K).VECI(J)
. . . . Next K
. . . . ECIVEL(J) = CINTRP(EPHT,EPHXYZ,DTREF)
. . . Next J
. . . Invoke SPHVEL(ECIPOS,ECIVEL,VSPH) to calculate the three
        spherical velocity components VSPH at DTREF;
. . . VRAD(NOUT) = Round(VSPH(1)/1000.0) [with unit conversion]
. . . VTHETA(NOUT) = Round(VSPH(2)/1000.0) [with unit conversion]
. . . VPHI(NOUT) = Round(VSPH(3)/1000.0) [with unit conversion]
```

```
. . {SOL_POS} [Calculate the solar celestial components, by
      interpolation and transformation]
. . . For J = 1 to 2
. . . . SOLX(J) = EPH(J+1).SOLECI(1)
. . . . SOLY(J) = EPH(J+1).SOLECI(2)
. . . . SOLZ(J) = EPH(J+1).SOLECI(3)
. . . Next J
. . . Invoke GCINTR(EPHT(2),SOLX,SOLY,SOLZ,DTREF,RLAT,RLON) to
        calculate the interpolated spherical coordinates RLAT,
        RLON from the sampled rectangular components (with proper
        angle limits);
. . . SOLRA(NOUT) = Round(100.0*RLON)
. . . SOLDEC(NOUT) = Round(100.0*RLAT)

. . [Initialize eclipse values, for possible reassignment]
. . ECLSTA(NOUT) = 0
. . ECLPCT(NOUT) = 0
. . [Determine the eclipse status, by comparison of entry and
      exit times]
. . For J = 1 to 4
. . . For K = 1 to 2
. . . . If (DTPENE(J,K) ≤ DTREF ≤ DTPENX(J,K)) Then[3]
. . . . . [This is at least a penumbral eclipse, so set
            percentage]
. . . . . ECLPCT(NOUT) = HDR(J).ECLIPSE(K)
. . . . . [Check for umbral occurrence]
. . . . . If (DTUMBE(J,K) ≤ DTREF ≤ DTUMBX(J,K)) Then[4]
. . . . . . ECLSTA(NOUT) = 2 [umbral]
. . . . . Else
. . . . . . ECLSTA(NOUT) = 1 [penumbral]
. . . . . End If
. . . . End If
. . . Next K
. . Next J

. . ATTP(NOUT) = CINTRP(EVTT,ATTPTCH,DTREF)
. . ATTY(NOUT) = CINTRP(EVTT,ATTYAW,DTREF)
. . ATTR(NOUT) = CINTRP(EVTT,ATTROLL,DTREF)
. . ROTP(NOUT) = CINTRP(EVTT,RATEPTCH,DTREF)
. . ROTY(NOUT) = CINTRP(EVTT,RATEYAW,DTREF)
. . ROTR(NOUT) = CINTRP(EVTT,RATEROLL,DTREF)

. . SMASS(NOUT) = (?)
```

---

[3] This time comparison may need to be generalized, depending upon the filler values used in HEADER.

[4] This time comparison may need to be generalized, depending upon the filler values used in HEADER.

```
. . CG1(NOUT) = (?)
. . CG2(NOUT) = (?)
. . CG3(NOUT) = (?)
. . I11(NOUT) = (?)
. . I22(NOUT) = (?)
. . I33(NOUT) = (?)
. . I12(NOUT) = (?)
. . I13(NOUT) = (?)
. . I23(NOUT) = (?)

. . [Pick the closest time sample for each thruster, and pack
      status into thruster word]
. . INEAR = NEARIDX(EVTT,DTREF)
. . For K = 1 to 4 [each thruster]
. . . THR(K) = ACS(INEAR).THRUST(K)
. . Next K
. . THRFLG(NOUT) = PACK(THR)

. . [Use the same closest time sample for each Torqrod, and pack
      status into Torqrod word]
. . For K = 1 to 3 [each Torqrod]
. . . TRQ(K) = TORQ(INEAR).ACT(K)
. . Next K
. . TRQ(4) = 0
. . TRQFLG(NOUT) = PACK(TRQ)

. . [Write Thruster/Torqrod report item if either is currently
      active]
. . If THRFLG(NOUT) ≠ 0 (?) or TRQFLG(NOUT) ≠ 0 (?) Then[5]
. . . Write MRGDN(NOUT), MRGTIM(NOUT)/10, ORBNUM(NOUT),
        ALT(NOUT), LEG(VRAD(NOUT)), (THR(I), I = 1 to 4),
        (TRQ(I), I = 1 to 3) to THRUST;
. . End If

. . [Write report for first sample after gap]
. . If RPTBEG = TRUE Then
. . . Write MRGDN(NOUT), MRGTIM(NOUT)/10, ORBNUM(NOUT),
        ALT(NOUT), LEG(VRAD(NOUT)) to LOG;
. . . RPTBEG = FALSE
. . End If
. . [Write merged data block to MERGE when full block is
      accumulated]
. . If NOUT ≥ MAXOUT Then
. . . Write NOUT, (MRGDATA(L), L = 1, NOUT) to MERGE;
. . . NOUT = 0
. . End If
```

---

[5] Need to verify activation values for thrusters and torqrods.

```
. . DTLAST = DTREF
. . ILAST = I(p)
. Next I
. [Determine which input sample to use for initiating the
    decimated sequence for the next input block]
. If ISEL > NSAMP Then
. . [The current filter block was skipped, so adjust starting
      index for the next block]
. . ISEL = ISEL - NSAMP
. Else
. . [Project next starting index based on last index used and
      requested decimation]
. . ISEL = ILAST + MDECIM
. End If
. Proceed from {GET_DATA};

{END_MERGE} [Write out partial block, and conclude processing]
. If NOUT > 0 Then
. . Write NOUT, (MRGDATA(L), L = 1, NOUT) to MERGE;
. . Close MERGE;
. End If
. Exit program;
```

Subroutines

```
LEG(VRAD)
  [Return indicator for orbital leg: +1 for upleg, -1 for
    downleg]
  VRAD = (R*4) radial component of satellite velocity, in
    spherical inertial coordinates
  LEG = (I*2) upleg/downleg indicator

LEG = Sign(1.0,VRAD) [1.0 takes sign of VRAD]
Return to calling routine;
```

```
NDAYS(IYR,IMON,IDAY)
  [Calculate the SETA day number for specified calendar date]
  IYR = (I*2) calendar date year
  IMON = (I*2) calendar date month number
  IDAY = (I*2) calendar date day of month
  NDAYS = (I*2) SETA day number
  See Function definition in Raw Data Unpacking or Raw Data
  Checking programs
```

```
GETEVT(EVT,EVTSTAT)
  [Acquire ACS data records from EVENT (see structure definition
  on page 26)]
  EVT = EVT HDR and ACS record
  EVTSTAT = (I*2) acquisition status value (0 for no errors)
Parameter value:
ACS = 25

EVTSTAT = 0 [initialization]

{READEVT} Read a (combined) EVTHDR and EVTDATA record, recording
  the read status as EVTSTAT;
. If EVTSTAT ≠ 0 Then
. . Return to calling routine;
. Else
. . If EVTHDR.IDPROC = ACS Then
. . . Return the combined EVTHDR and EVTACS record (as EVT), and
        EVTSTAT to calling routine;
. . Else
. . . Proceed from {READEVT};
. . End If
. End If
```

```
GETHDR(HDR,HDRSTAT)
  [Acquire data records from HEADER (see Agency Data Tape Header
  definition on page 28)]
  HDR = HDR record
  HDRSTAT = (I*2) acquisition status value (0 for no errors)
Initial values:
INDEX = 1
NREVS = 0

HDRSTAT = 0 [initialization]

{READHDR} [Read a new HEADER record only when the current source
  record has been completely utilized]
. If INDEX > NREVS Then
. . Read a HDRDATA record, recording the read status as HDRSTAT;
. . INDEX = 1
. End If
. If HDRSTAT = 0 Then
. . [Store the number of available revolutions]
. . NREVS = NREV[6]
. . [Map the relevant subset of the HDRDATA information into the
      HDR record]
. . HDR.BT = BTASCN(INDEX)
. . HDR.REVNUM = KREV(INDEX)
. . For J = 1 to 2
. . . HDR.ECLIPSE(J) = ECLOBS(INDEX,J)
. . . HDR.ENTPEN(J) = PENBEG(INDEX,J)
. . . HDR.ENTUMB(J) = UMBBEG(INDEX,J)
. . . HDR.LVUMB(J) = UMBEND(INDEX,J)
. . . HDR.LVPEN(J) = PENEND(INDEX,J)
. . Next J
. . INDEX = INDEX + 1
. End If
. Return the HDR record and HDRSTAT to calling routine;
```

```
CVTBT(BT)
  [Convert the standard ADT packed "BINARY" time format
  (YYMMDDHHMMSS) into a SETA day number and fractional day value]
  BT = (6×I*1) six byte record structure:
    YY = (I*1) year since 1900
    MM = (I*1) month number
    DD = (I*1) day of month
    HH = (I*1) hour
    MN = (I*1) minute
    SS = (I*1) second
  CVTBT = (R*8) SETA day number and fractional day
```

[6] Is NREV = 0 a possibility?

```
If BT ≠ -1 Then[7]
. CVTBT = NDAYS(YY,MM,IDD) + (3600*HH + 60*MN + SS)/86400.0
Else
. CVTBT = 0.0
End If
Return to calling routine;
```

```
NEARIDX(TIMES,TREF)
  [Return the index value nearest in time to TREF, from the 4
  time tags TIMES]
  TIMES(4) = (R*8) day number and fractional day time tags
  TREF = (R*8) day number and fractional day for selection
  NEARIDX = (I*2) index value

NEARIDX = 1
TDIF = Abs(TREF - TIMES(1))
For I = 2 to 4
. TCOMP = Abs(TREF - TIMES(I))
. If TCOMP < TDIF Then
. . TDIF = TCOMP
. . NEARIDX = I
. End If
Next I
Return to calling routine;
```

```
PACK(INTS)
  [Pack 4 single-byte integers into 4 half-bytes for a single 2-
  byte value]
  INTS(4) = (I*1) integer values between 0 and 15
  PACK = (I*2) packed value

Initial values:
MASK = 15 [half-byte mask]

PACK = IAnd(ZExt(INTS(4)), MASK)(q)
For I = 3 to 1 by -1
. PACK = IShft(PACK,4)(r)
. PACK = IOr(PACK,IAnd(ZExt(INTS(I)), MASK))(s)
Next I
Return to calling routine;
```

---

[7] The fill value for BT is not determined, and the logical test may need to be performed on the individual bytes.

```
CINTRP(X,Y,X0)
  [Perform cubic interpolation over four (X,Y) pairs to obtain Y
    value at X0 (as CINTRP)]
  X(4) = (R*8) independent variable for interpolation
  Y(4) = (R*8) dependent variable for interpolation
  X0 = (R*8) selected value requiring dependent value

D1 = X0 - X(1)
D2 = X0 - X(2)
D3 = X0 - X(3)
D4 = X0 - X(4)

X12 = X(1) - X(2)
X13 = X(1) - X(3)
X14 = X(1) - X(4)
X23 = X(2) - X(3)
X24 = X(2) - X(4)
X34 = X(3) - X(4)

CINTRP = Y(1)*(D2/X12)*(D3/X13)*(D4/X14)
  - Y(2)*(D1/X12)*(D3/X23)*(D4/X24)
  + Y(3)*(D1/X13)*(D2/X23)*(D4/X34)
  - Y(4)*(D1/X14)*(D2/X24)*(D3/X34)

Return to calling routine;
```

```
SPHVEL(RECT,VRECT,VSPH)
  [Convert rectangular velocity components VRECT to spherical
    velocity components VSPH, at rectangular coordinate position
    RECT]
  RECT(3) = (R) rectangular position coordinates
  VRECT(3) = (R) rectangular velocity coordinates
  VSPH(3) = (R) spherical velocity coordinates
  See Density and Winds Check-out program for this routine.
```

```
GCINTS(T,LAT,LON,T0,LAT0,LON0)
  [Perform interpolation along an arc for an intermediate
    position, for spherical coordinate inputs]
  T(2) = (R*8) initial and final times, for limits of arc (as day
    and fraction of day)
  LAT(2) = (R*8) initial and final "latitudinal" positions along
    arc (degrees)
  LON(2) = (R*8) initial and final "longitudinal" position along
    arc (degrees)
  T0 = (R*8) specified time for intermediate position
  LAT0 = (R*8) intermediate "latitudinal" position at specified
    time (degrees)
  LON0 = (R*8) intermediate "longitudinal" position at specified
```

```
    time (degrees)

For I = 1 to 2
. X(I) = CosD(LAT(I))*CosD(LON(I))
. Y(I) = CosD(LAT(I))*SinD(LON(I))
. Z(I) = SinD(LAT(I))
Next I

Invoke GCINTR(T,X,Y,Z,T0,LAT0,LON0) to interpolate the sampled
  rectangular coordinates X, Y, Z to calculate the spherical
  coordinate angles at time T0;
Return to calling routine;
```

```
GCINTR(T,X,Y,Z,T0,LAT0,LON0)
  [Perform interpolation along an arc for an intermediate
    position, for rectangular coordinate inputs]
  T(2) = (R*8) initial and final times, for limits of arc (as day
    and fraction of day)
  X(2) = (R*8) initial and final X-coordinate positions along arc
  Y(2) = (R*8) initial and final Y-coordinate position along arc
  Z(2) = (R*8) initial and final Z-coordinate position along arc
  T0 = (R*8) specified time for intermediate position
  LAT0 = (R*8) intermediate "latitudinal" position at specified
    time (degrees)
  LON0 = (R*8) intermediate "longitudinal" position at specified
    time (degrees)

[Calculate the full and partial time intervals]
DT21 = T(2) - T(1)
DT01 = T0 - T(1)

[Calculate the dot product and angular separation for the end
  points]
PROJ = X(1)*X(2) + Y(1)*Y(2) + Z(1)*Z(2)
OMEGA = ACos(PROJ)

[Calculate the angular separation for the interpolated point, and
  the associated interpolation coefficients]
DELTA = (DT01/DT21)*OMEGA
A = Sin(DELTA)/Abs(Sin(OMEGA))
B = Cos(DELTA) - A*PROJ

[Calculate the rectangular coordinates of the interpolated point]
X0 = A*X(2) + B*X(1)
Y0 = A*Y(2) + B*Y(1)
Z0 = A*Z(2) + B*Z(1)
```

```
[Calculate the spherical angle coordinates of the interpolated
  point]
LAT0 = ASinD(Z0)(t)
LON0 = ATan2D(Y0,X0)(u)

Return to calling routine;
```

SETA/ADS Software Development
Ephemeris and Attitude Merge Program

Definitions and Notes

a. EPHT = composite day number/time for ephemeris data samples

b. EVTT = composite day number/time for event data samples

c. HDRT = composite day number/time for header data samples

d. NOUT = number of output samples for MERGE block, and index of current sample

e. DTPREV = composite day number/time for last sample of previous input filtered data block

f. DTLAST = composite day number/time for most recent sample of output merged data block

g. DTFIRST = composite day number/time for first sample in input filtered data block

h. ISEL = index of first filtered acceleration sample to be used as a decimated merge sample

i. RPTBEG = logical variable, set TRUE to report values for beginning of data segment after time gap

j. DTNEXT = composite day number/time for projected time of next decimated merge sample

k. DTREF = composite day number/time for current input filtered sample and output merged sample

l. DTPENE is day number and fraction for penumbra entry time

m. DTUMBE is day number and fraction for umbra entry time

n. DTUMBX is day number and fraction for umbra exit time

o. DTPENX is day number and fraction for penumbra exit time

p. ILAST = index of last sample in input filtered data block used for output merge sample

q. IAnd is a VAX extension for bitwise AND
ZExt is a VAX extension for zero-fill leading bit extension of a word size

r. IShft is a VAX extension for bit-pattern shifting (positive leftward)

s. IOr is a VAX extension for bitwise OR

t. ASinD is a VAX extension for arcsine in degrees

u. ATan2D is a VAX extension for two-argument arctangent in degrees

SETA/ADS Software Development
Ephemeris and Attitude Merge Program

EPH Structure

Item	Description	Type
1. IYY	Time Tag Year (since 1900)	I*1
2. IMM	Time Tag Month	I*1
3. IDD	Time Tag Day (of Month)	I*1
4. IHH	Time Tag Hour	I*1
5. IMN	Time Tag Minute	I*1
6. ISS	Time Tag Second	I*1
7. MJDAY	Modified Julian Day (standard Julian Day - 2400000.5)	I*4
8. UTMSEC	Universal Time in milliseconds	I*4
9. ECI(1) = XECI	Satellite X-position in meters (ECI, mean equinox of date)	I*4
10. ECI(2) = YECI	Satellite Y-position in meters (ECI, mean equinox of date)	I*4
11. ECI(3) = ZECI	Satellite Z-position in meters (ECI, mean equinox of date)	I*4
12. VECI(1) = VXECI	Satellite X-velocity in millimeters/second (ECI, mean equinox of date)	I*4
13. VECI(2) = VYECI	Satellite Y-velocity in millimeters/second (ECI, mean equinox of date)	I*4
14. VECI(3) = VZECI	Satellite Z-velocity in millimeters/second (ECI, mean equinox of date)	I*4
15. RMAG	Radius vector magnitude to satellite (from earth center) in meters	I*4
16. ALT	Satellite altitude in meters, from reference ellipsoid	I*4
17. GLAT	Geocentric latitude in micro-degrees	I*4
18. GLON	Geocentric longitude in micro-degrees, positive East	I*4
19. VMAG	Velocity vector magnitude in millimeters/second	I*4
20. LT	Local time in hours times 10^6	I*4
21. GMR	Satellite radial position in earth-centered dipole geomagnetic coordinates (in $10^3\times$EMR)	I*4
22. GMLAT	Satellite latitude in earth-centered dipole geomagnetic coordinates (micro-degrees)	I*4
23. GMLON	Satellite longitude in earth-centered dipole geomagnetic coordinates (micro-degrees, positive East from meridian containing South geographic pole)	I*4
24. SMR	Satellite radial position in earth eccentric dipole solar magnetic coordinates (in $10^3\times$EMR)	I*4
25. SMLAT	Satellite latitude in earth eccentric dipole solar magnetic coordinates (micro-degrees)	I*4
26. SMLT	Satellite local time in earth eccentric dipole solar magnetic coordinates (hours times 10^6)	I*4
27. GSMR	Satellite radial position in earth eccentric dipole solar magnetospheric coordinates (in $10^3\times$EMR)	I*4
28. GSMLAT	Satellite latitude in earth eccentric dipole solar magnetospheric coordinates (micro-degrees)	I*4
29. GSMLT	Satellite local time in earth eccentric dipole solar magnetospheric coordinates (hours times 10^6)	I*4
30. BMAG	Magnitude of model magnetic field in milli-gammas	I*4
31. BXECI	Model magnetic field ECI X-component in pico-Tesla	I*4
32. BYECI	Model magnetic field ECI Y-component in pico-Tesla	I*4
33. BZECI	Model magnetic field ECI Z-component in pico-Tesla	I*4
34. GMLT	Geomagnetic local time in hours times 10^6	I*4
35. SOLANG	Geocentric angle between satellite and sun in micro-degrees	I*4
36. INVLAT	L-shell invariant latitude parameter in micro-degrees	I*4
37. BFILATN	Geocentric latitude in micro-degrees for 100 km northern field line trace intercept	I*4
38. BFILONN	Geocentric longitude (+E) in micro-degrees for 100 km northern field line trace intercept	I*4
39. BFILATS	Geocentric latitude in micro-degrees for 100 km southern field line trace intercept	I*4
40. BFILONS	Geocentric longitude (+E) in micro-degrees for 100 km southern field line trace intercept	I*4
41. LSHELL	L-shell parameter in 10^6 times EMR	I*4
42. BMIN	Minimum field strength along current magnetic field line in pico-Tesla	I*4
43. BMLAT	Geocentric latitude for minimum magnetic field strength location along current field line (micro-degrees)	I*4
44. BMLON	Geocentric longitude for minimum magnetic field strength location along current field line (micro-degrees)	I*4
45. BMRAD	Geocentric radial coordinate for minimum magnetic field strength location along current field line (meters)	I*4
46. BCNJLAT	Conjugate point geocentric latitude in micro-degrees	I*4

EPH Structure
(continued)

Item	Description	Type
47. BCNJLON	Conjugate point geocentric longitude in micro-degrees	I*4
48. BCNJRAD	Conjugate point geocentric radial coordinate in meters	I*4
49. SOLECI(1) = SOLECIX	Solar X-coordinate in kilometers (ECI)	I*4
50. SOLECI(2) = SOLECIY	Solar Y-coordinate in kilometers (ECI)	I*4
51. SOLECI(3) = SOLECIZ	Solar Z-coordinate in kilometers (ECI)	I*4
52. LUNECIX	Lunar X-coordinate in kilometers (ECI)	I*4
53. LUNECIY	Lunar Y-coordinate in kilometers (ECI)	I*4
54. LUNECIZ	Lunar Z-coordinate in kilometers (ECI)	I*4
55. GRA	Right ascension of Greenwich mean sidereal time in micro-degrees	I*4
56. BFIMAGN	Magnetic field magnitude in pico-Tesla for 100 km northern field line trace intercept	I*4
57. BFIMAGS	Magnetic field magnitude in pico-Tesla for 100 km southern field line trace intercept	I*4
58. BMECIX	Dipole field moment X-component in pico-Tesla (ECI)	I*4
59. BMECIY	Dipole field moment Y-component in pico-Tesla (ECI)	I*4
60. BMECIZ	Dipole field moment Z-component in pico-Tesla (ECI)	I*4
61. BOECIX	Eccentric dipole offset X-component in meters (ECI)	I*4
62. BOECIY	Eccentric dipole offset Y-component in meters (ECI)	I*4
63. BOECIZ	Eccentric dipole offset Z-component in meters (ECI)	I*4

MRGHDR Structure

Item	Description	Type
1. EXPID	Experiment Identifier ('SETA-5')	C*8
2. DTYPE	Data Type ('MERGE ')	C*8
3. SAMPRT	Nominal Data Sampling Rate (Hz) (≡ 10)	C*8
4. DECIM	Decimation Factor for Data Segment	C*4
5. ASCALE	Scale Factor for Acceleration Data (to micro-Gees)	C*8
6. BEGYR	Year Number for Beginning of Data Segment	C*4
7. BEGMON	Month Number for Beginning of Data Segment	C*2
8. BEGDAY	Day of Month Number for Beginning of Data Segment	C*2
9. BEGDN	SETA Day Number for Beginning of Data Segment	C*4
10. GENDN	SETA Day Number for Processing of Raw Data Segment	C*4
11. FILTDN	SETA Day Number for Filtering of Data Segment	C*4
12. MRGDN	SETA Day Number for Merging of Data Segment	C*4
13. DENDN	SETA Day Number for Density/Wind Processing (Blank)	C*4
14. TGAP	Time Gap Allowance (Seconds) for Interpolating	C*4
15. WPTHRX	Wild Point Threshold: X-Acceleration (micro-Gees)	C*8
16. WPTHRY	Wild Point Threshold: Y-Acceleration (micro-Gees)	C*8
17. WPTHRZ	Wild Point Threshold: Z-Acceleration (micro-Gees)	C*8
18. WPTHRT	Wild Point Threshold: Temperature (^{o}C)	C*8
19. WPEDX	Wild Point Editing Flag: X-Acceleration	C*1
20. WPEDY	Wild Point Editing Flag: Y-Acceleration	C*1
21. WPEDZ	Wild Point Editing Flag: Z-Acceleration	C*1
22. WPEDT	Wild Point Editing Flag: Temperature	C*1
23. FILTX(K), K=1,..,4	X-Acceleration Filter Parameters	C*8
24. FILTY(K), K=1,..,4	Y-Acceleration Filter Parameters	C*8
25. FILTZ(K), K=1,..,4	Z-Acceleration Filter Parameters	C*8
26. FILTT(K), K=1,..,4	Temperature Filter Parameters	C*8
27. AREF	Reference Area for Drag Coefficient (m^2) (Blank)	C*8
28. POS1	Accelerometer Location: STEP-X Coordinate (mm)	C*8
29. POS2	Accelerometer Location: STEP-Y Coordinate (mm)	C*8
30. POS3	Accelerometer Location: STEP-Z Coordinate (mm)	C*8
31. CALOPT	Calculation Option for Densities and Winds (Blank)	C*2

MRGDATA Structure

Item	Description	Type
1. MRGDN	SETA Day Number for Data Sample	I*2
2. MRGTIM	Time of Day for Data Sample, in tenths of seconds	I*4
3. ACCX	SETA-X (STEP -Y) Filtered Acceleration Scaled Units	I*2
4. ACCY	SETA-Y (STEP -Z) Filtered Acceleration Scaled Units	I*2
5. ACCZ	SETA-Z (STEP +X) Filtered Acceleration Scaled Units	I*2
6. TEMP	SETA Filtered Temperature, in Degrees	I*2
7. RNGFLG	Accelerometer Range Flags (Packed)	I*2
8. ORBNUM	Orbit Number	I*2
9. ALT	Vehicle Altitude (m)	I*4
10. LAT	Vehicle Latitude, Positive North (hundredths of a degree)	I*2
11. LON	Vehicle Longitude, Positive East (hundredths of a degree)	I*2
12. RAD	Local Orbit Radius (m)	I*4
13. GMLAT	Geomagnetic Latitude, Positive North (hundredths of a degree)	I*2
14. GMLON	Geomagnetic Longitude, Positive East (hundredths of a degree)	I*2
15. GMLT	Geomagnetic Local Time (tenths of seconds)	I*2
16. VRAD	Vehicle Radial Velocity (Inertial Coordinates, m/sec)	I*2
17. VTHETA	Vehicle Latitudinal Velocity, Positive South (Inertial Coordinates, m/sec)	I*2
18. VPHI	Vehicle Longitudinal Velocity, Positive East (Inertial Coordinates, m/sec)	I*2
19. SOLRA	Solar Right Ascension (degrees)	I*2
20. SOLDEC	Solar Declination (degrees)	I*2
21. ECLSTA	Eclipse Status	I*1
22. ECLPCT	Peak Percentage Eclipse	I*1
23. ATTP	Pitch Attitude Angle (arc-min)	I*2
24. ATTY	Yaw Attitude Angle (arc-min)	I*2
25. ATTR	Roll Attitude Angle (arc-min)	I*2
26. ROTP	Pitch Attitude Rate (arc-sec/sec)	I*2
27. ROTY	Yaw Attitude Rate (arc-sec/sec)	I*2
28. ROTR	Roll Attitude Rate (arc-sec/sec)	I*2
29. SMASS	Vehicle Mass (kg)	I*2
30. CG1	Vehicle Center-of-Mass STEP-X Coordinate (mm)	I*2
31. CG2	Vehicle Center-of-Mass STEP-Y Coordinate (mm)	I*2
32. CG3	Vehicle Center-of-Mass STEP-Z Coordinate (mm)	I*2
33. I11	Vehicle Moment of Inertia: I_{xx} (kg-cm^2) (STEP coordinates)	I*4
34. I22	Vehicle Moment of Inertia: I_{yy} (kg-cm^2) (STEP coordinates)	I*4
35. I33	Vehicle Moment of Inertia: I_{zz} (kg-cm^2) (STEP coordinates)	I*4
36. I12	Vehicle Moment of Inertia: I_{xy} (kg-cm^2) (STEP coordinates)	I*4
37. I13	Vehicle Moment of Inertia: I_{xz} (kg-cm^2) (STEP coordinates)	I*4
38. I23	Vehicle Moment of Inertia: I_{yz} (kg-cm^2) (STEP coordinates)	I*4
39. THRFLG	Thruster Firing Flags (Packed)	I*2
40. TRQFLG	Torqrod Excitation Flags (Packed)	I*2

APPENDIX F - Bias Determination Program

SETA/ADS Software Development
Bias Determination Program

Files:
MERGE - SETA merged data in PL format
SOLAR - history file of solar flux and geomagnetic activity
BIAS - SETA bias file
LOG - log file of bias, acceleration, and model values

Parameters:
LimTbl = 1000 [maximum number of disjoint data segments to be used for composite duration for bias determination; used as dimension of TBeg, TEnd, NRec]

Overview:
1. Scan through MERGE data for condition of bias determination;
2. Select appropriate data segment for bias determination, using backward search through indexing table for Upleg and forward search through indexing table for Downleg, to preferentially select higher altitudes;
3. Backtrack through the MERGE data to acquire the data samples for processing;
4. If necessary (below 500 km), calculate model densities and use attitude information to determine drag accelerations for each accelerometer axis;
5. List and store measured minus model accelerations as bias values.

Notes:
I. Based on the current orbital scenario, the accelerometer is assumed to be operational for more than 10 minutes. If intermittent operation is implemented, then the EVENT file will need to be accessed to determine the turn-on time for the accelerometer from the command history.
II. The orbit number assigned to the bias values will correspond to the associated time of day.
III. A MERGE data record is a logical block of up to 64 samples.

{SET_VAL} [Value initialization]
. ALTLIM = 500 [altitude limit below which to use model values for density]
. TREQ = 500 [requested time interval for bias determination, in seconds]
. TMIN = 60 [minimum acceptable time interval for bias determination, in seconds]
. TPrev = 0 [day and fraction for previous MERGE sample]
. KSAMP = 0 [data sample number within MERGE record]
. APDT = 0 [day and fraction for current 3-hour Ap]

{START_UP}
. Open MERGE data file for input;
. If error on open Then terminate program, with error message;

```
. Open SOLAR data file for input;
. If error on open Then
. . Issue error warning that model densities will not be
      calculated;
. . Set ALTTHR = ALTLIM [use 500 km altitude threshold instead of
      300 km];
. Else
. . Set ALTTHR = 300 [300 km altitude threshold];
. End If
. Open BIAS data file for output [may need APPEND provisions
    here];
. Open LOG listing file for output;

{INIT} [Initialize file acquisitions and processing flags]
. NUMREC = 0 [number of MERGE records acquired for current bias
    calculation]
. LEG = 0 [upleg/downleg designation]
. Read and store MERGE data header;
. [Store the accelerometer offsets, converting from millimeters
    to meters]
. . AccPos(1) = 0.001*POS1
. . AccPos(2) = 0.001*POS2
. . AccPos(3) = 0.001*POS3
. TGAP = (1.5*DECIM/SAMPRT)/86400.0 [time gap definition, as
    fraction of day]
. NBMin = TMIN/(DECIM/SAMPRT) + 1 [minimum number of samples
    required for bias calculation]
. [Acquire the initial records from the solar activity file]
. For K = 1 to 5
. . Read MYr, Mon, MDay, SOLAR(K).Vals from SOLAR parameter
      file;(a)
. . If end-of-file or error on read Then
. . . Report error to user and LOG;
. . . If K > 1 Then
. . . . Report SOLAR.DT(K-1) to user and LOG, as last time
          successfully acquired;
. . . End If
. . . Issue error warning that model densities will not be
        calculated;
. . . Set ALTTHR = ALTLIM [use 500 km altitude threshold instead
        of 300 km];
. . End If
. . SOLAR(K).DT = NDAYS(MYr, Mon, MDay)
. Next K

{CHECK_DATA} [Scan MERGE for bias determination conditions]
. Invoke FETCH(KSAMP, SAMPLE, NUMREC, EOF) to read a single
    sample from a MERGE data record;
. If EOF is TRUE Then proceed to {END};
. [Set values for segment index table]
```

```
. . IxTbl = 1
. . TCurr = SAMPLE.MRGDN + SAMPLE.MRGTIM/864000.0 [sample time as
      day and fraction]
. . TBeg(1) = TCurr [beginning day/time for continuous data
      segment]
. . NRec(1) = NUMREC [current data record number]
. [Convert MERGE altitudes from meters to kilometers for
    comparison]
. If SAMPLE.ALT/1000.0 < ALTTHR Then
. . Proceed to {SCAN};
. Else
. . Set LEG = Sign(1.0,SAMPLE.VRAD) [+1 for upleg, -1 for
      downleg];
. . If LEG = +1 Then
. . . BackSrch = TRUE(b)
. . . Proceed to {SCAN_TO_DOWNLEG};
. . Else
. . . BackSrch = FALSE
. . . Proceed to {SCAN_TO_ALTTHR};
. . End If
. End If

{SCAN} [Scan through MERGE data for acceptable altitude]
. TPrev = TCurr [save day/time for reference, as "previous"
    sample]
. VrPrev = SAMPLE.VRAD [save radial velocity for reference, as
    "previous" sample]
. Invoke FETCH(KSAMP, SAMPLE, NUMREC, EOF) to read a single
    sample from a MERGE data record;
. If EOF is TRUE Then proceed to {END};
. [Define TCurr for next use in setting TPrev]
. TCurr = SAMPLE.MRGDN + SAMPLE.MRGTIM/864000.0
. If SAMPLE.ALT/1000.0 ≥ ALTTHR Then
. . IxTbl = 1
. . TBeg(IxTbl) = TCurr
. . NRec(IxTbl) = NUMREC
. . BackSrch = TRUE
. . Proceed to SCAN_TO_DOWNLEG;
. Else
. . Proceed from {SCAN};
. End If

{SCAN_TO_DOWNLEG} [Scan through (implicitly upleg) samples until
  a downleg sample is encountered]
. TPrev = TCurr [save day/time for reference, as "previous"
    sample]
. VrPrev = SAMPLE.VRAD
. Invoke FETCH(KSAMP, SAMPLE, NUMREC, EOF) to read a single
    sample from a MERGE data record;
. If EOF is TRUE Then
```

```
. . TEnd(IxTbl) = TPrev
. . Proceed to {BIAS};
. Else
. . TCurr = SAMPLE.MRGDN + SAMPLE.MRGTIM/864000.0
. . [Check for time gap]
. . If (TCurr - TPrev) > TGAP Then
. . . TEnd(IxTbl) = TPrev
. . . Increment IxTbl;
. . . If IxTbl > LimTbl Then
. . . . Report error condition to user:
. . . . . "Too many gaps in data; IxTbl =", IxTbl, "LimTbl = ",
          LimTbl, "TCurr = ", TCurr
. . . . Set IxTbl = LimTbl;
. . . End If
. . . TBeg(IxTbl) = TCurr
. . . NRec(IxTbl) = NUMREC
. . . GAP = TRUE
. . Else
. . . GAP = FALSE
. . End If
. . LEG = Sign(1.0,SAMPLE.VRAD)
. . If LEG = +1 Then
. . . Proceed from {SCAN_TO_DOWNLEG};
. . Else
. . . If GAP is TRUE Then
. . . . TEnd(IxTbl) = TCurr
. . . . Decrement IxTbl [Ignore isolated point after gap];
. . . . Proceed to {BIAS};
. . . Else
. . . . [Refine the apogee time estimate by linear interpolation
         (radial velocity is zero at apogee)]
. . . . TApog = (TPrev*SAMPLE.VRAD - TCurr*VrPrev)/(SAMPLE.VRAD -
         VrPrev)
. . . . [Stopping conditions are:
         equal time durations about apogee,
         continuous segment of requested duration]
. . . . TStop = Max(TApog + TREQ/2, TBeg(IxTbl) + TREQ)
. . . . Proceed to {AP_BIAS};
. . . End If
. . End If
. End If

{SCAN_TO_ALTTHR} [Scan through (implicitly downleg) samples until
 the threshold altitude is encountered]
. TPrev = TCurr [save day/time for reference, as "previous"
   sample]
. Invoke FETCH(KSAMP, SAMPLE, NUMREC, EOF) to read a single
   sample from a MERGE data record;
. If EOF is TRUE Then
. . TEnd(IxTbl) = TPrev
```

```
. . Proceed to {BIAS};
. Else
. . TCurr = SAMPLE.MRGDN + SAMPLE.MRGTIM/864000.0
. . [Check for time gap]
. . If (TCurr - TPrev) > TGAP Then
. . . TEnd(IxTbl) = TPrev
. . . Increment IxTbl;
. . . If IxTbl > LimTbl Then
. . . . Report error condition to user:
. . . . . "Too many gaps in data; IxTbl =", IxTbl, "LimTbl = ",
          LimTbl, "TCurr = ", TCurr
. . . . Set IxTbl = LimTbl;
. . . End If
. . . TBeg(IxTbl) = TCurr
. . . NRec(IxTbl) = NUMREC
. . . GAP = TRUE
. . Else
. . . GAP = FALSE
. . End If
. . If SAMPLE.ALT/1000.0 ≥ ALTTHR Then
. . . Proceed from {SCAN_TO_ALTTHR};
. . Else
. . . If GAP is TRUE Then
. . . . TEnd(IxTbl) = TCurr
. . . . Decrement IxTbl [Ignore isolated point after gap];
. . . Else
. . . . [Terminate the data segment table at the previous sample]
. . . . TEnd(IxTbl) = TPrev
. . . End If
. . . Proceed to {BIAS};
. . End If
. End If

{AP_BIAS} [Prepare for collecting data samples bracketing apogee]
. TPrev = TCurr [save day/time for reference, as "previous"
    sample]
. Invoke FETCH(KSAMP, SAMPLE, NUMREC, EOF) to read a single
    sample from a MERGE data record;
. If EOF is TRUE Then
. . TEnd(IxTbl) = TPrev
. . Proceed to {BIAS};
. Else
. . TCurr = SAMPLE.MRGDN + SAMPLE.MRGTIM/864000.0
. . [Check for time gap]
. . If (TCurr - TPrev) > TGAP Then
. . . TEnd(IxTbl) = TPrev
. . . Proceed to {BIAS};
. . Else
. . . If TCurr ≥ TStop Then
. . . . TEnd(IxTbl) = TCurr
```

```
. . . . Proceed to {BIAS};
. . . Else
. . . . Proceed from {AP_BIAS};
. . End If
. End If

{BIAS} [Select the actual time range to be used for the bias
  determination]
. Cum_T = 0 [Cumulative time of candidate data segments]
. If BackSrch is TRUE Then
. . [Set loop limits to examine successive segments backwards to
     accumulate requested time duration, if possible]
. . IStart = IxTbl
. . IFinish = 1
. . IInc = -1
. Else
. . [Set loop limits to examine successive segments forwards to
     accumulate requested time duration, if possible]
. . IStart = 1
. . IFinish = IxTbl
. . IInc = 1
. End If
. For I = IStart to IFinish by IInc:
. . If (TEnd(I) - TBeg(I)) ≥ TREQ/86400.0 Then
. . . [One of the segments encompasses the requested time
       duration]
. . . If BackSrch is TRUE Then
. . . . T1 = TEnd(I) - TREQ/86400.0 [Start time of data segment
         for bias determination]
. . . . T2 = TEnd(I) [End time of data segment for bias
         determination]
. . . Else
. . . . T1 = TBeg(I) [Start time of data segment for bias
         determination]
. . . . T2 = TBeg(I) + TREQ/86400.0 [End time of data segment for
         bias determination]
. . . End If
. . . IBeg = I [Index in segment table for starting time]
. . . NGaps = 0 [Number of gaps in composite interval]
. . . Proceed to {SET_CALC};
. . Else
. . . If Cum_T < TREQ Then
. . . . [Update the cumulative time duration for this data
         segment]
. . . . Cum_T = Cum_T + 86400*(TEnd(I) - TBeg(I))
. . . . ICum = I
. . . End If
. . End If
. Next I
. If Cum_T ≥ TREQ Then
```

```
. . [A sufficient time duration has been accumulated (with no
      individual segment longer than the requested duration)]
. . If BackSrch is TRUE Then
. . . T1 = TBeg(ICum) + (Cum_T - TREQ)/86400.0
. . . T2 = TEnd(IxTbl) [End time of composite segments]
. . . IBeg = ICum
. . Else
. . . T1 = TBeg(1) [Start time of composite segments]
. . . T2 = TEnd(ICum) - (Cum_T - TREQ)/86400.0
. . . IBeg = 1
. . End If
. . NGaps = ICum - 1
. . Proceed to {SET_CALC};
. Else
. . [Check whether the minimal duration request has been
      satisfied]
. . If Cum_T ≥ TMIN Then
. . . T1 = TBeg(1)
. . . T2 = TEnd(IxTbl)
. . . IBeg = 1
. . . Proceed to {SET_CALC};
. . Else
. . . [Skip to next bias determination opportunity if conditions
        are not satisfied]
. . . Proceed from {CHECK_DATA};
. . End If
. End If

{SET_CALC} [Initialize for bias calculations]
. Last_Rec = NUMREC [Save the current record number]
. Invoke BACK_SPACE(NRec(IBeg),NUMREC) to return to MERGE record
    number NREC(IBeg), which is the start of the current bias
    determination segment;
. NBSamp = 0 [Number of bias samples used]
. NBEst = (T2 - T1)/(DECIM/SAMPRT) + 1 + NGaps [Estimated number
    of samples for selected duration]
. NMSamp = 0 [Number of model samples used]
. AveTemp = 0 [Cumulative/average accelerometer temperature for
    selected duration]
. For J = 1 to 3
. . AveBias(J) = 0 [Cumulative/average bias for each
      accelerometer axis: X, Y, Z]
. . AveAcc(J) = 0 [Cumulative/average accelerations reported for
      each axis]
. . AveMdl(J) = 0 [Cumulative/average model drag accelerations
      calculated for each axis, not including "drag-free" region]
. Next J
```

```
{CALC} [Calculate the bias values, referencing the atmospheric
  model if necessary]
. Invoke FETCH(KSAMP, SAMPLE, NUMREC, EOF) to read a single
    sample from a MERGE data record;
. If EOF is TRUE Then
. . Invoke BIAS_RPT(NBSamp, AveBias, AveAcc, AveTemp, NMSamp,
      AveMdl, TMid, OrbMid, AltMid) to store and report the
      cumulative and average results;
. . Proceed to {END};
. Else
. . TCurr = SAMPLE.MRGDN + SAMPLE.MRGTIM/864000.0
. . If TCurr > T2 Then
. . . Invoke BIAS_RPT(NBSamp, AveBias, AveAcc, AveTemp, NMSamp,
        AveMdl, TMid, OrbMid, AltMid) to store and report the
        cumulative and average results;
. . . Proceed to {RESET};
. . Else
. . . If TCurr ≥ T1 Then
. . . . [This is part of the selected data segments]
. . . . If SAMPLE.ALT/1000.0 < ALTLIM Then
. . . . . [Need to calculate model density and modeled
            accelerations]
. . . . . {SOLAR_BRACKET} [For selected input sample, find the
            bracketing SOLAR records (4 preceding, 1 succeeding),
            for averaging][8]
. . . . . If TCurr < SOLAR(4).DT Then
. . . . . . [There is a problem with the SOLAR coverage, which
              starts within a day after the accelerometer data or
              has a gap, or a time reversal has occurred in the
              accelerometer data]
. . . . . . Report TCurr, SOLAR.DT to user and LOG, with error
              message about SOLAR coverage;
. . . . . . Invoke MDL_TERM(ALTTHR, ALTLIM, NBMin, NBSamp, NRec,
              NumRec, IBeg, T1, AveBias, AveAcc, AveTemp, NMSamp,
              AveMdl, TMid, OrbMid, AltMid) to reset the altitude
              threshold and determine the appropriate action for
              the current bias segment;
. . . . . . Proceed to {RESET};
. . . . . Else
. . . . . . {GET_SOLAR} [Read SOLAR until bracketing times are
              acquired]
. . . . . . If TCurr ≥ SOLAR(5).DT Then
. . . . . . . [Shuffle reference samples to prepare for new
                acquisition]
. . . . . . . For K = 1 to 4
. . . . . . . . SOLAR(K) = SOLAR(K+1)
. . . . . . . Next K
```

[8] See similar procedure for DENSITY design.

```
. . . . . . . . . [Note: Nine elements are in each daily AP record;
                    Two elements are in FLUX record]
. . . . . . . . . Read MYr, Mon, MDay, SOLAR(5).Vals from SOLAR
                    parameter file;
. . . . . . . . . If end-of-file or error on read Then
. . . . . . . . . . Report error to user and LOG;
. . . . . . . . . . Report TCurr, SOLAR(4).DT to user and LOG, as
                      last time successfully acquired from SOLAR
                      activity file;
. . . . . . . . . . Invoke MDL_TERM(ALTTHR, ALTLIM, NBMin, NBSamp,
                      NRec, NumRec, IBeg, T1, AveBias, AveAcc,
                      AveTemp, NMSamp, AveMdl, TMid, OrbMid, AltMid)
                      to reset the altitude threshold and determine
                      the appropriate action for the current bias
                      segment;
. . . . . . . . . . Proceed to {RESET};
. . . . . . . . . End If
. . . . . . . . . SOLAR(5).DT = NDAYS(MYr, Mon, MDay)
. . . . . . . . . Proceed from {GET_SOLAR};
. . . . . . . . End If
. . . . . . . End If
. . . . . . . If 24.0*ABS(TCurr - APDT) > 1.5 Then
. . . . . . . . {SEL_AP} [Set the appropriate records and elements
                  for the current and prior 3-hour AP values][9]
. . . . . . . . IREC = 4
. . . . . . . . IELEM = 8*Frac(TCurr) + 1
. . . . . . . . [Need to be careful how solar activity values are
                  time-referenced (center time for 3-hour intervals)]
. . . . . . . . APDT = SOLAR(IREC).DT + (IELEM - 0.5)/8.0
. . . . . . . . AP(1) = SOLAR(IREC).APDAILY
. . . . . . . . For K = 2 to 5
. . . . . . . . . AP(K) = SOLAR(IREC).AP(IELEM)
. . . . . . . . . IELEM = IELEM - 1
. . . . . . . . . If IELEM = 0 Then
. . . . . . . . . . IELEM = 8
. . . . . . . . . . IREC = IREC - 1
. . . . . . . . . End If
. . . . . . . . Next K
. . . . . . . . [Average the prior 16 AP values, in two groups]
. . . . . . . . For L = 1 to 2
. . . . . . . . . APAVE = 0
. . . . . . . . . For K = 1 to 8
. . . . . . . . . . APAVE = APAVE + SOLAR(IREC).AP(IELEM)
. . . . . . . . . . IELEM = IELEM - 1
. . . . . . . . . . If IELEM = 0 Then
. . . . . . . . . . . IELEM = 8
. . . . . . . . . . . IREC = IREC - 1
```

[9] See corresponding segment in DENSITY design.

```
. . . . . . . . . . If IREC ≤ 0 Then
. . . . . . . . . . . Report "Problem with average AP
                        determination", SOLAR(4).DT, "Prior Record
                        = ",L, "Element =",K to LOG
. . . . . . . . . . . [This is not considered a fatal error]
. . . . . . . . . . End If
. . . . . . . . . End If
. . . . . . . . Next K
. . . . . . . . AP(5+L) = APAVE/8.0
. . . . . . . Next L
. . . . . . . [Solar flux for previous day and average to current
                day, with scaling]
. . . . . . . FLUX = SOLAR(3).FLUX
. . . . . . . FLUXAV = SOLAR(4).FLUXAV
. . . . . . End If
. . . . . . Invoke ATM_MODEL(SAMPLE, AP, FLUX, FLUXAV, Dens, WtMol,
              AmbTemp) to calculate the MSIS model density at the
              required position (latitude, longitude, local time,
              altitude);
. . . . . Else
. . . . . . [Assume no atmospheric drag above ALTLIM]
. . . . . . Dens = 0
. . . . . End If
. . . . . Invoke DRAGS(SAMPLE, AccPos, Dens, ACC, ACC_MDL) to:
            (a) calculate the corrected measured accelerations
                (ACC(3)), accounting for rotational and gravity
                gradient effects;
            (b) calculate the expected drag components (ACC_MDL(3))
                for each accelerometer axis, using the appropriate
                attitude, satellite mass, and drag coefficients;
. . . . . Increment NBSamp;
. . . . . If NBSamp = NBEst/2 Then
. . . . . . OrbMid = SAMPLE.ORBNUM [Orbit number tag for output]
. . . . . . TMid = TCurr [Date/time tag for output]
. . . . . . AltMid = SAMPLE.ALT [Altitude tag for output]
. . . . . End If
. . . . . AveTemp = AveTemp + SAMPLE.TEMP
. . . . . For J = 1 to 3
. . . . . . AveAcc(J) = AveAcc(J) + ACC(J)
. . . . . . AveBias(J) = AveBias(J) + ACC(J) - ACC_MDL(J)
. . . . . Next J
. . . . . If SAMPLE.ALT/1000.0 < ALTLIM Then
. . . . . . Increment NMSamp;
. . . . . . For J = 1 to 3
. . . . . . . AveMdl(J) = AveMdl(J) + ACC_MDL(J)
. . . . . . Next J
. . . . . End If
. . . . End If
. . . End If
. . . Proceed from {CALC};
```

```
. End If

{RESET} [Re-position MERGE file at last record previously scanned
  for bias segments]
. If NUMREC < Last_Rec Then
. . KSAMP = 0 [set sample index to trigger next record
     acquisition]
. . Invoke FETCH(KSAMP, SAMPLE, NUMREC, EOF) to acquire a new
     record from MERGE;
. . If EOF is TRUE Then
. . . Proceed to {END};
. . Else
. . . Proceed from {RESET};
. . End If
. End If

{SET_POS} [Re-position MERGE file at end of sequence previously
  scanned for bias segments]
. Invoke FETCH(KSAMP, SAMPLE, NUMREC, EOF) to read a single
    sample from a MERGE data record;
. TCurr = SAMPLE.MRGDN + SAMPLE.MRGTIM/864000.0
. If TCurr ≥ TEnd(IxTbl) Then
. . Proceed from {CHECK_DATA};
. Else
. . Proceed from {SET_POS};
. End If

{END} [Conclude processing]
. Issue conclusion report (with day and time of last data sample)
    to LOG;
. Close files:
. . MERGE
. . SOLAR
. . BIAS
. . LOG
. Exit;
```

Subroutines

```
FETCH(KSAMP, SAMPLE, NUMREC, EOF)
  KSAMP = (I*2) next sample number in data record to be acquired
    (input/output);
  SAMPLE = (MRGDATA structure) MERGE file data sample (output);
  NUMREC = (I*4) record number in MERGE file (input/output);
  EOF = (L*2) logical flag, set TRUE for end-of-file or error
    (output);

Parameters:
  MaxSamp = 64 [maximum number of samples for MERGE record;
    dimension of MRG_BLK]

. If KSamp = 0 Then
. . Invoke GET_BLK(NSAMP, MRG_BLK, NUMREC, EOF) to acquire a
      single record from MERGE;
. . If EOF is TRUE Then
. . . Return to calling routine;
. . End If
. . KSAMP = 1
. End If
. [Acquire all elements of a single MERGE sample as a SAMPLE
    record]
. Set SAMPLE = MRG_BLK(KSamp);
. Increment KSamp;
. If KSamp > NSAMP Then
. . KSamp = 0
. End If
. Return to calling routine;
```

```
GET_BLK(NSAMP, MRG_BLK, NUMREC, EOF)
  NSAMP = (I*4) number of time samples in MERGE data record
    (output);
  MRG_BLK = (MRGDATA structure) MERGE file data block (output);
  NUMREC = (I*4) record number in MERGE file (input/output);
  EOF = (L*2) logical flag, set TRUE for end-of-file or error
    (output);

Parameters:
  MaxSamp = 64 [maximum number of samples for MERGE record;
    dimension of MRG_BLK]

. Read data record from MERGE file;
. If end-of-file or error Then
. . Report end-of-file or error, with NUMREC, to user;
. . Set EOF = TRUE;
. End If
. Increment NUMREC;
```

```
. Return to calling routine;
```

```
BACK_SPACE(NRDest, NRCurr)
[Re-position the MERGE file, to recover the data values for the
  bias determination or re-scan]
  NRDest = record number to return to (input);
  NRCurr = current record number (input/output);

. NRBack = NRCurr - NRDest
. Backspace MERGE by NRBack records;
. NRCurr = NRCurr - NRBack
. KSAMP = 1

. Return to calling routine;
```

```
BIAS_RPT(NBSamp, AveBias, AveAcc, AveTemp, NMSamp, AveMdl, TMid,
  OrbMid, AltMid)
  NBSamp = (I*2) number of bias samples used;
  AveBias(3) = (R*4) cumulative/average bias for each
    accelerometer axis: X, Y, Z;
  AveAcc(3) = (R*4) cumulative/average accelerations reported for
    each axis;
  AveTemp = (R*4) cumulative/average accelerometer temperature
    for selected duration;
  NMSamp = (I*2) number of model samples used;
  AveMdl(3) = (R*4) cumulative/average model drag accelerations
    calculated for each axis, not including "drag-free" region;
  TMid = (R*8) date/time tag for report;
  OrbMid = (I*2) orbit number tag for report;
  AltMid = (I*4) altitude tag for report;

. [Calculate the required averages]
. AveTemp = AveTemp/NBSamp
. For J = 1 to 3
. . AveBias(J) = AveBias(J)/NBSamp
. . AveAcc(J) = AveAcc(J)/NBSamp
. . If NMSamp > 0 Then AveMdl(J) = AveMdl(J)/NMSamp
. Next J

. BiasDN = Int(TMid)
. BiasTm = (TMid - BiasDN)*86400
. BiasAlt = AltMid/1000.0

. Write OrbMid, BiasDN, BiasTm, NBSamp, BiasAlt, (AveBias(J), J =
    1 to 3), AveTemp to BIAS;
. Write OrbMid, BiasDN, BiasTm, NBSamp, BiasAlt, (AveBias(J), J =
    1 to 3), AveTemp to LOG;
. Write (AveAcc(J), J = 1 to 3) to LOG;
```

. Write NMSamp, (AveMdl(J), J = 1 to 3) to LOG;

. Return to calling routine;

MDL_TERM(ALTTHR, ALTLIM, NBMin, NBSamp, NRec, NUMREC, IBeg, T1, AveBias, AveAcc, AveTemp, NMSamp, AveMdl, TMid, OrbMid, AltMid)

ALTTHR = (R*4) altitude threshold (km) for bias determination (input/output);
ALTLIM = (R*4) altitude limit (km) below which model density is required (input);
NBMin = (I*4) minimum number of samples required for bias calculation (input);
NBSamp = (I*2) number of bias samples used (input);
NRec(LimTbl) = (I*4) array for MERGE record numbers used for bias calculation (input);
NUMREC = (I*4) current record number in MERGE file (input/output);
IBeg = (I*2) index in NRec for start of bias sequence (input);
T1 = (R*8) composite day/time for beginning of bias calculation sequence (input);
AveBias(3) = (R*4) cumulative/average bias for each accelerometer axis: X, Y, Z (input/output);
AveAcc(3) = (R*4) cumulative/average accelerations reported for each axis (input/output);
AveTemp = (R*4) cumulative/average accelerometer temperature for selected duration (input/output);
NMSamp = (I*2) number of model samples used (input);
AveMdl(3) = (R*4) cumulative/average model drag accelerations calculated for each axis, not including "drag-free" region (input/output);
TMid = (R*8) date/time tag for report (input/output);
OrbMid = (I*2) orbit number tag for report (input/output);
AltMid = (I*4) altitude tag for report (input/output);

. Issue error warning that model densities will no longer be calculated;
. Set ALTTHR = ALTLIM [use 500 km altitude threshold instead of 300 km];
. [Need to re-check conditions for valid bias calculation, and compensate for possible premature termination]
. If NBSamp ≥ NBMin Then
. . [At least the minimal number of samples has been acquired, so perform a bias calculation using the available cumulatives, but backtrack (if necessary) to obtain the proper median values]
. . If NBSamp ≠ NBEst Then
. . . [The bias segment retrieval ended prematurely, so backtrack to obtain the proper median values]

```
. . . NMid = NBSamp/2 [New median point, based on number of
        samples actually acquired]
. . . Invoke BACK_SPACE(NRec(IBeg),NUMREC) to return to MERGE
        record number NREC(IBeg), which contains the start of the
        current bias determination segment;
. . . {FIND_START} [Find the starting sample used for the current
        bias segment]
. . . . Invoke FETCH(KSAMP, SAMPLE, NUMREC, EOF) to read a single
          sample from a MERGE data record;
. . . . TCurr = SAMPLE.MRGDN + SAMPLE.MRGTIM/864000.0
. . . . If TCurr < T1 Then proceed from {FIND_START};
. . . For I = 1 to NMid-1
. . . . Invoke FETCH(KSAMP, SAMPLE, NUMREC, EOF) to read a single
          sample from a MERGE data record;
. . . Next I
. . . [This should be the new median sample]
. . . OrbMid = SAMPLE.ORBNUM
. . . TMid = SAMPLE.MRGDN + SAMPLE.MRGTIM/864000.0
. . . AltMid = SAMPLE.ALT
. . End If
. . Invoke BIAS_RPT(NBSamp, AveBias, AveAcc, AveTemp, NMSamp,
      AveMdl, TMid, OrbMid, AltMid) to store and report the
      cumulative and average results;
. . Proceed from {RESET};
. Else
. . If BackSrch is TRUE Then
. . . [This is an upleg (re-)scan, which will be resumed as a
        survey with a higher ALTTHR]
. . . Proceed from {SCAN};
. . Else
. . . [This is a downleg (re-)scan, which could possibly be
        salvaged by selecting from a regenerated index table]
. . . Invoke BACK_SPACE(NRec(1),NUMREC) to return to MERGE record
        number NRec(1), which is the start of the current bias
        determination sequence;
. . . Proceed from {SCAN_TO_ALTTHR};
. . End If
. End If
```

ATM_MODEL(SAMPLE, AP, F107, F107A, Dens, WtMol, ThermT)

SAMPLE = MRGDATA structure for a data sample (input);
AP(7) = (R*4) array of 3-hour and average A_p values for model calculation (input);
F107 = (R*4) daily solar flux value for model (input);
F107A = (R*4) average solar flux value for model (input);
Dens = (R*4) mass density (g/cm^3) from model (output);
WtMol = (R*4) mean molecular weight (AMU) for model (output);
ThermT = (R*4) ambient gas temperature (degrees Kelvin) from

model (output);

Parameters:
AMU = 1.66053 x 10^{-24} [Atomic Mass Unit, in grams]

. [Define the variables required by the MSIS-90 routine]
. [Calculate the day-of-year, on a 365-day basis]
. IYD = Int(Mod(SAMPLE.DATDN-0.5, 365.25) + 1)[c]
. SEC = SAMPLE.DATTIM/10.0[d]
. ALT = SAMPLE.ALT/1000.0[e]
. GLAT = SAMPLE.LAT/100.0[f]
. GLONG = SAMPLE.LON/100.0[g]
. STL = SEC/3600.0 + GLONG/15.0[h]
. Invoke MSIS-90 routine GTD6(IYD, SEC, ALT, GLAT, GLONG, STL, F107A, F107, AP, D, T) to obtain mass and number densities (D) and local and exospheric temperatures (T);
. [Calculate the mean molecular weight, in Atomic Mass Units, from the components]
. WtMol = (D(6)/(D(1) + D(2) + D(3) + D(4) + D(5) + D(7) + D(8)))/AMU[i]
. Dens = D(6)
. ThermT = T(2)

. Return to calling routine;

DRAGS(SAMPLE, AccPos, Dens, ACC, ACC_MDL)
[Correct the measured accelerations and calculate the model accelerations]
Input:
SAMPLE = (MRGDATA structure) MERGE file data sample;
AccPos(3) = satellite coordinates for accelerometer (m);
Dens = density from atmosphere model (g/cm^3)

Output:
ACC(3) = measured accelerations, corrected for satellite rotation and gravity gradient;
ACC_MDL(3) = accelerations from model density, with satellite orientation effects;

Global input:
ARef = reference frontal area for satellite (m^2) [from DrgCoef]
ASCALE = scale factor for accelerometer data conversion to micro-G's [from MERGE header]

Local parameters:
GAcc = 9.8 [nominal gravitational acceleration at earth's surface (m/sec^2)]
GConv = 9.8×10^{-6} [conversion from micro-G's to m/sec^2]
Omega = 7.292123517×10^{-5} (= $2\pi/86164$) [sidereal rotation rate

```
    of earth, in radians/sec]
  RadVec(3) = (0, 0, -1.0) [local outward unit radius vector, in
    nominal satellite coordinates]
  REarth = 6377569 (= √(GMe/g)) [nominal earth radius for
    gravitational acceleration (m)]

  Conv = 10^3/GConv [conversion constant in drag equation]

. [Obtain the rotation rates in radians/sec, in satellite
    coordinates]
. Rot1 = RadCnv*SAMPLE.ROTR/3600.0(j)
. Rot2 = RadCnv*SAMPLE.ROTP/3600.0(k)
. Rot3 = RadCnv*SAMPLE.ROTY/3600.0(l)
. RotSq = Rot1**2 + Rot2**2 + Rot3**2
. [Determine the accelerometer offset from the center-of-mass,
    for satellite coordinates in meters]
. CMass(1) = 0.001*SAMPLE.CG1
. CMass(2) = 0.001*SAMPLE.CG2
. CMass(3) = 0.001*SAMPLE.CG3
. For I = 1 to 3
. . Offset(I) = AccPos(I) - CMass(I)
. Next I
. RotPrj = Rot1*OffSet(1) + Rot2*OffSet(2) + Rot3*OffSet(3)
. [Determine the gravity gradient corrections, to first order,
    assuming the attitude angles are with respect to the Local
    Vertical/Ram Direction system]
. OrbRad = SAMPLE.RAD
. Pitch = RadCnv*(SAMPLE.PITCH/60.0)
. Yaw = RadCnv*(SAMPLE.YAW/60.0)
. Roll = RadCnv*(SAMPLE.ROLL/60.0)
. Invoke EulTrns(Pitch,Yaw,Roll,XForm) to compute the
    transformation matrix for the specified attitude;
. Invoke IMSL:MURRV(3,3,XForm,3,3,RadVec,1,3,VertRef) to
    transform (using matrix multiplication by the transpose of
    XForm) the local unit radius vector (RadVec) in the nominal
    satellite coordinate system (vertical and ram alignment) to a
    reference vertical (VertRef) in current satellite
    coordinates;
. RadPrj = OffSet(1)*VertRef(1) + OffSet(2)*VertRef(2) +
    OffSet(3)*VertRef(3)
. For K = 1 to 3
. . GGrad(K) = GAcc*(REarth/OrbRad)**2*(OffSet(K) -
      3*RadPrj*VertRef(K))/OrbRad
. Next K
. [Calculate drag acceleration (in micro-G's) in satellite
    coordinates, correcting for rotational effects and gravity
    gradient]
. ACC(1) = ASCALE*SAMPLE.ACCZ + ((RotPrj*Rot1 - RotSq*OffSet(1))
    + GGrad(1))/GConv
. ACC(2) = -ASCALE*SAMPLE.ACCX - ((RotPrj*Rot2 - RotSq*OffSet(2))
```

```
    + GGrad(2))/GConv
. ACC(3) = -ASCALE*SAMPLE.ACCY - ((RotPrj*Rot3 - RotSq*OffSet(3))
    + GGrad(3))/GConv

. If Dens ≠ 0 Then
. . [Calculate the acceleration expected from the model density,
      using the expected bulk gas flow at the satellite, based on
      the satellite motion and the earth's rotation, but
      neglecting winds]
. . VTrnsv = Sqrt(SAMPLE.VTHETA**2 + SAMPLE.VPHI**2)(m)
. . VCoRot = SAMPLE.RAD*Omega*CosD(SAMPLE.LAT/100.0)(n)
. . ECVel(1) = VCoRot*SAMPLE.VPHI/VTrnsv - VTrnsv
. . ECVel(2) = -VCoRot*SAMPLE.VTHETA/VTrnsv
. . ECVel(3) = SAMPLE.VRAD
. . Invoke IMSL:MURRV(3,3,XForm,3,3,ECVel,1,3,Vel) to transform
      (using matrix multiplication by the transpose of XForm) the
      local earth-centered velocity vector (ECVel) in the nominal
      satellite coordinate system (vertical and ram alignment) to
      a reference velocity (Vel) in current satellite
      coordinates;
. . [Calculate the drag coefficient components]
. . Invoke DrgCoef(Vel, WtMol, ThermT, CMass, CD, Torque)
. . [Calculate the model drag acceleration, including conversion
      factor for Dens (g/cm^3 to kg/m^3) and conversion to micro-
      G's]
. . VMagSq = Vel(1)**2 + Vel(2)**2 + Vel(3)**2
. . For I = 1 to 3
. . . ACC_MDL(I) = 0.5*Conv*Dens*VMagSq*CD(I)*ARef/SAMPLE.SMASS
. . Next I
. Else
. . [No density, so no model drag]
. . For I = 1 to 3
. . . ACC_MDL(I) = 0
. . Next I
. End If
. Return to calling routine;
```

SETA/ADS Software Development
Bias Determination Program

Definitions and Notes

a. MYr = 4-digit year for solar activity/flux data
Mon = 2-digit month for solar activity/flux data
MDay = 2-digit day-of-month for solar activity/flux data
Vals Structure:
APDaily = daily A_p value
AP(8) = successive 3-hour A_p values for the day
Flux = daily $F_{10.7}$ flux for the day
Flux90 = average $F_{10.7}$ flux over the previous three months (90 days)

b. BackSrch = flag for backward or forward searching through time segment table, set TRUE for backward search

c. IYD = day-of-year, from 1 to 365; (need additional mod(IYD-1,365) + 1 to strictly enforce 365-day limit)

d. SEC = Universal Time in seconds

e. ALT = altitude in kilometers

f. GLAT = geocentric latitude in degrees (should be geodetic for MSIS-90)

g. GLONG = geocentric longitude in degrees, positive East (should be geodetic for MSIS-90)

h. STL = local apparent solar time in hours

i. WtMol = mean molecular weight, in atomic mass units

j. Rot1 = rotation rate about satellite X-axis, in radians/sec (Roll)

k. Rot2 = rotation rate about satellite Y-axis, in radians/sec (Pitch)

l. Rot3 = rotation rate about satellite Z-axis, in radians/sec (Yaw)

m. VTrnsv = total transverse (horizontal) satellite velocity

n. VCoRot = co-rotating thermospheric velocity at satellite altitude

APPENDIX G - Density and Winds Calculation Program

SETA/ADS Software Development
Density and Winds Calculation Program

Files:
SOURCE - SETA merged data in PL format, or SETA density data in PL format, used as input
BIAS - SETA bias file
SOLAR - history file of solar flux and geomagnetic activity
DENSITY - SETA density data in PL format, used as output
LOG - log file of processing status

Parameters:
MAXINP = 64 [maximum number of samples in an input data block (SOURCE)]
MAXOUT = 48 [maximum number of samples in an output data block (DENSITY)]

Physical and conversion constants:
AMU = 1.66053×10^{-24} [Atomic Mass Unit, in grams]
GAcc = 9.8 [nominal gravitational acceleration at earth's surface, in m/sec^2]
GConv = 9.8×10^{-6} [conversion from micro-G's to m/sec^2]
Omega = 7.292123517×10^{-5} (= $2\pi/86164$) [sidereal rotation rate of earth, in radians/sec]
RadVec = (0, 0, -1.0) [local outward unit radius vector, in nominal satellite coordinates]
REarth = 6377569 (= $\sqrt{(GM_e/g)}$) [nominal earth radius, in meters, for gravitational acceleration]

Overview:
1. Determine type of source data (Merge or Density) by reading Data Type from SOURCE header;
2. Acquire data values from SOURCE into interim storage (SRC);
3. Determine bracketing solar activity, ADMS, and CADS samples, and assign required quantities to time of data sample (by interpolation or nearest occurrence);
4. Invoke the bias reporting routine to obtain the bias values for the accelerometer axes at the time of the data sample, using either a global (or quasi-global) fit to the bias file values or an interpolation between the bias file values;
5. Calculate the density and wind values according to the specified analysis option;
6. Transfer processed data to output file, writing data to DENSITY when complete output blocks are accumulated.

{SET_VALS} [Initialize data values]
. MASS = 48[a]
. NOUT = 0[b]
. APDT = 0[c]
. WindVel(I) = 0, I = 1 to 3[d]

```
{OPTS} [Obtain user specifications]
. Read processing options from file or terminal, or retain
    defaults, in parentheses:
. . OptReq (= 0) [calculation option for density and wind
      determination (to be selected consistent with available
      data sources)]
. IReq = OptReq/10(e)

{INIT} [Initialization]
. Open SOURCE data file for input;
. If error on open Then terminate program, with error message;
. {READ_HDR} [Read and store header information from SOURCE input
    file]
. . Read header items from file SOURCE: [See data format
      descriptions]
. . EXPID
. . DTYPE
. . SAMPRT
. . DECIM
. . ASCALE
. . BEGYR
. . BEGMON
. . BEGDAY
. . BEGDN
. . GENDN
. . FILTDN
. . MRGDN
. . DENDN
. . TGAP
. . WPTHRX
. . WPTHRY
. . WPTHRZ
. . WPTHRT
. . WPEDX
. . WPEDY
. . WPEDZ
. . WPEDT
. . FILTX(K), K = 1 to 4
. . FILTY(K), K = 1 to 4
. . FILTZ(K), K = 1 to 4
. . FILTT(K), K = 1 to 4
. . AREF
. . POS1
. . POS2
. . POS3
. . CALOPT
. [Report date of data]
. Write 'Data Source', EXPID, DTYPE, BEGYR, BEGMON, BEGDAY to LOG
. Store input data type for further use:
. . InType = DTYPE
```

```
. Open ADMS data file for input;
. If error on open Then
. . [Verify that ADMS source file is not required for processing]
. . If IReq = 1 and InType = 'MERGE' Then
. . . [The ADMS data is required but not available]
. . . Terminate program, with error message;
. . End If
. Else
. . GetADMS = TRUE
. End If

. Open CADS data file for input;
. If error on open Then
. . [Verify that CADS source file is not required for processing]
. . If IReq = 2 and InType = 'MERGE' Then
. . . [The CADS data is required but not available]
. . . Terminate program, with error message;
. . End If
. Else
. . GetCADS = TRUE
. End If

. Open SOLAR parameter file for input;
. If error on open Then terminate program, with error message;

. Open BIAS history file for input;
. [Note: This may not be required if a parametric fit is used, or
    the initialization may be handled within the Bias routine]
. If error on open Then terminate program, with error message;

. Open DENSITY data file for output;
. Open LOG listing file for output;

. Initialize variables from data acquisition:
. . If GetADMS = TRUE Then
. . . {INIT_ADMS} Read initial time and values from ADMS file
        [see ADMS structure definition on page 35];
. . End If

. . If GetCADS = TRUE Then
. . . {INIT_CADS} Read initial time and values from CADS file
        [see CADS structure definition on page 36];
. . End If

. . [Acquire the initial records from the solar activity file]
. . For K = 1 to 5
. . . Read MYr, Mon, MDay, SOLAR(K).Vals from SOLAR parameter
        file;(f)
. . . If end-of-file or error on read Then
. . . . Report error to user and LOG;
```

```
. . . . If K > 1 Then
. . . . . Report SOLAR.DT(K-1) to user and LOG, as last time
            successfully acquired;
. . . . End If
. . . . Proceed to {END_DENSITY};
. . . End If
. . . SOLAR(K).DT = NDAYS(MYr, Mon, MDay)
. . Next K

{WRITE_HDR} [Write header for output density data file]
. Obtain processing date (current date) from system, as IPYR
    [year], IPMON [month], IPDAY [day of month];
. DENDN = NDAYS(IPYR,IPMON,IPDAY)
. DTYPE = 'DENSITY '
. AREF = value from Drag routine (/Params/RefArea)
. CALOPT = OptReq
. Write header items to file DENSITY [See data format
    descriptions]

{GET_DATA} [Acquire merge or density data block]
. If InType = 'DENSITY ' Then
. . [Read the data using the full DENSITY structure]
. . Read NSAMP, (SRC(L), L = 1, NSAMP) from SOURCE [See data
      format descriptions];
. Else
. . [Read the data using the MERGE substructure]
. . Read NSAMP, (SRC(L).MERGE, L = 1, NSAMP) from SOURCE [See
      data format descriptions];
. End If
. If end-of-file on read Then
. . [This should not happen during a data block without error];
. . Report end-of-file to user;
. . If NOUT > 0 Then write OUTDATA(NOUT).DATDN,
      OUTDATA(NOUT).DATTIM/10, OUTDATA(NOUT).ORBNUM,
      OUTDATA(NOUT).ALT to LOG;
. . Proceed to {END_DENSITY};
. Else If error on read Then
. . Report error (with error number) to user;
. . If NOUT > 0 Then write OUTDATA(NOUT).DATDN,
      OUTDATA(NOUT).DATTIM/10, OUTDATA(NOUT).ORBNUM,
      OUTDATA(NOUT).ALT to LOG;
. . Report error (with error number) to LOG;
. . Proceed to {END_DENSITY};
. End If

{PROCESS} [Process the current block of data]
. For I = 1 to NSAMP
. . DTREF = SRC(I).DATDN + SRC(I).DATTIM/864000.0

. . {INST_MRG} [Merge auxiliary instrument data into
```

```
      accelerometer data]
. . {ADMS_BRACKET} [For selected input sample, find the
      bracketing ADMS records, for interpolation]
. . If GetADMS = TRUE Then
. . . If DTREF < ADMS.DT(2) Then
. . . . [There is a problem with the ADMS coverage, which starts
          within a minute (?) after the accelerometer data or has
          a gap, or a time reversal has occurred in the
          accelerometer data]
. . . . Report DTREF, ADMS.DT to user and LOG, with error message
          about ADMS coverage;
. . . . Proceed to {END_DENSITY};
. . . Else If DTREF ≥ ADMS.DT(3) Then
. . . . {GET_ADMS} [Read ADMS until bracketing times are
          acquired]
. . . . [Shuffle reference samples to prepare for new
          acquisition]
   .
   .
   .
. . . . If end-of-file or error on read Then
. . . . . Report error to user and LOG;
. . . . . Report ADMS.DT(3) to user and LOG, as last time
            successfully acquired;
. . . . . Proceed to {END_DENSITY};
. . . . End If
. . . . If DTREF ≥ ADMS.DT(3) Then proceed from {GET_ADMS};
. . . End If
. . End If

. . {CADS_BRACKET} [For selected input sample, find the
      bracketing CADS records, for interpolation]
. . If GetCADS = TRUE Then
. . . If DTREF < CADS.DT(2) Then
. . . . [There is a problem with the CADS coverage, which starts
          within a minute (?) after the accelerometer data or has
          a gap, or a time reversal has occurred in the
          accelerometer data]
. . . . Report DTREF, CADS.DT to user and LOG, with error message
          about CADS coverage;
. . . . Proceed to {END_DENSITY};
. . . Else If DTREF ≥ CADS.DT(3) Then
. . . . {GET_CADS} [Read CADS until bracketing times are
          acquired]
. . . . [Shuffle reference samples to prepare for new
          acquisition]
   .
   .
   .
. . . . If end-of-file or error on read Then
```

```
. . . . . . Report error to user and LOG;
. . . . . . Report CADS.DT(3) to user and LOG, as last time
              successfully acquired;
. . . . . . Proceed to {END_DENSITY};
. . . . . End If
. . . . . If DTREF ≥ CADS.DT(3) Then proceed from {GET_CADS};
. . . . End If
. . . End If

. . {SOLAR_BRACKET} [For selected input sample, find the
      bracketing SOLAR records (4 preceding, 1 succeeding), for
      averaging]
. . If DTREF < SOLAR(4).DT Then
. . . [There is a problem with the SOLAR coverage, which starts
        within a day after the accelerometer data or has a gap,
        or a time reversal has occurred in the accelerometer
        data]
. . . Report DTREF, SOLAR.DT to user and LOG, with error message
        about SOLAR coverage;
. . . Proceed to {END_DENSITY};
. . Else If DTREF ≥ SOLAR(5).DT Then
. . . {GET_SOLAR} [Read SOLAR until bracketing times are
        acquired]
. . . [Shuffle reference samples to prepare for new acquisition]
. . . For K = 1 to 4
. . . . SOLAR(K) = SOLAR(K+1)
. . . Next K
. . . [Note: Nine elements are in each daily AP record; Two
        elements are in FLUX record]
. . . Read MYr, Mon, MDay, SOLAR(5).Vals from SOLAR parameter
        file;
. . . If end-of-file or error on read Then
. . . . Report error to user and LOG;
. . . . Report SOLAR(4).DT to user and LOG, as last time
          successfully acquired;
. . . . Proceed to {END_DENSITY};
. . . End If
. . . SOLAR(5).DT = NDAYS(MYr, Mon, MDay)
. . . If DTREF ≥ SOLAR(5).DT Then proceed from {GET_SOLAR};
. . End If

. . [A time bracket exists or has been generated for each
      reference file, so interpolate or match data items]
. . NOUT = NOUT + 1

. . {TAG_DATA} [Provide ADMS, CADS, solar activity, and bias
      information for selected sample]
. . Transfer MERGE substructure from input to output for the
      current sample:
. . . OUTDATA(NOUT).MERGE = SRC(I).MERGE
```

```
. . If GetADMS = TRUE Then
. . . [Interpolate for ADMS values, converting to storage units
        as necessary]
. . . OUTDATA(NOUT).ADEN = Round(10^15×LINTRP(ADMS(2).DT,
        ADMS(3).DT, ADMS(2).DENS, ADMS(3).DENS, DTREF))
. . . OUTDATA(NOUT).AWT = Round(1000×LINTRP(ADMS(2).DT,
        ADMS(3).DT, ADMS(2).MMWt, ADMS(3).MMWt, DTREF))
. . Else If InType = 'DENSITY ' Then
. . . [Fetch the values from the source density data]
. . . OUTDATA(NOUT).ADEN = SRC(I).ADEN
. . . OUTDATA(NOUT).AWT = SRC(I).AWT
. . Else
. . . [No values are available, so insure zeroes are stored]
. . . OUTDATA(NOUT).ADEN = 0
. . . OUTDATA(NOUT).AWT = 0
. . End If
. . If GetCADS = TRUE Then
. . . [Interpolate for CADS values, converting to storage units
        as necessary]
. . . OUTDATA(NOUT).CDEN = Round(10^15×LINTRP(CADS(2).DT,
        CADS(3).DT, CADS(2).DENS, CADS(3).DENS, DTREF))
. . . OUTDATA(NOUT).CWT = Round(1000×LINTRP(CADS(2).DT,
        CADS(3).DT, CADS(2).MMWt, CADS(3).MMWt, DTREF))
. . . OUTDATA(NOUT).CTEMP = Round(LINTRP(CADS(2).DT, CADS(3).DT,
        CADS(2).Temp, CADS(3).Temp, DTREF))
. . . OUTDATA(NOUT).CWIND = Round(LINTRP(CADS(2).DT, CADS(3).DT,
        CADS(2).Wind, CADS(3).Wind, DTREF))[10]
. . Else If InType = 'DENSITY ' Then
. . . [Fetch the values from the source density data]
. . . OUTDATA(NOUT).CDEN = SRC(I).CDEN
. . . OUTDATA(NOUT).CWT = SRC(I).CWT
. . . OUTDATA(NOUT).CTEMP = SRC(I).CTEMP
. . . OUTDATA(NOUT).CWIND = SRC(I).CWIND
. . Else
. . . [No values are available, so insure zeroes are stored]
. . . OUTDATA(NOUT).CDEN = 0
. . . OUTDATA(NOUT).CWT = 0
. . . OUTDATA(NOUT).CTEMP = 0
. . . OUTDATA(NOUT).CWIND = 0
. . End If
. . If 24.0*ABS(DTREF - APDT) > 1.5 Then
. . . {SEL_AP} [Set the appropriate records and elements for the
        current and prior 3-hour AP values]
. . . IREC = 4
. . . IELEM = 8*Frac(DTREF) + 1
. . . [Need to be careful how solar activity values are time-
```

[10] A unit conversion may be required for the CADS wind value.

```
          referenced (center time for 3-hour intervals)]
. . . APDT = SOLAR(IREC).DT + (IELEM - 0.5)/8.0
. . . AP(1) = SOLAR(IREC).APDAILY
. . . For K = 2 to 5
. . . . AP(K) = SOLAR(IREC).AP(IELEM)
. . . . IELEM = IELEM - 1
. . . . If IELEM = 0 Then
. . . . . IELEM = 8
. . . . . IREC = IREC - 1
. . . . End If
. . . Next K
. . . [Average the prior 16 AP values, in two groups]
. . . For L = 1 to 2
. . . . APAVE = 0
. . . . For K = 1 to 8
. . . . . APAVE = APAVE + SOLAR(IREC).AP(IELEM)
. . . . . IELEM = IELEM - 1
. . . . . If IELEM = 0 Then
. . . . . . IELEM = 8
. . . . . . IREC = IREC - 1
. . . . . . If IREC ≤ 0 Then
. . . . . . . Report "Problem with average AP determination",
                SOLAR(4).DT, "Prior Record = ",L, "Element =",K
                to LOG
. . . . . . . [This is not considered a fatal error]
. . . . . . End If
. . . . . End If
. . . . Next K
. . . . AP(5+L) = APAVE/8.0
. . . Next L
. . End If
. . For K = 1 to 7
. . . [Store the AP values for later use and output, with
        scaling]
. . . OUTDATA(NOUT).AP(K) = Round(10*AP(K))(g)
. . Next K
. . [Solar flux for previous day and average to current day, with
      scaling]
. . OUTDATA(NOUT).FLUXPR = Round(10*SOLAR(3).FLUX)
. . OUTDATA(NOUT).FLUXAV = Round(10*SOLAR(4).FLUXAV)
. . Invoke GetBias(DTREF, Bias) to obtain the three bias values
      at date/time DTREF;
. . OUTDATA(NOUT).BIASX = Round(Bias(1)/ASCALE)
. . OUTDATA(NOUT).BIASY = Round(Bias(2)/ASCALE)
. . OUTDATA(NOUT).BIASZ = Round(Bias(3)/ASCALE)

. . {MSIS} [Define the variables required by the MSIS-90 routine]
. . [Calculate the day-of-year, on a 365-day basis]
. . IYD = Int(Mod(OUTDATA(NOUT).DATDN-0.5, 365.25) + 1)(h)
. . SEC = OUTDATA(NOUT).DATTIM/10.0(i)
```

```
. . ALT = OUTDATA(NOUT).ALT/1000.0[j]
. . GLAT = OUTDATA(NOUT).LAT/100.0[k]
. . GLONG = OUTDATA(NOUT).LON/100.0[l]
. . STL = SEC/3600.0 + GLON/15.0[m]
. . [Use stored values, for roundoff consistency]
. . F107A = OUTDATA(NOUT).FLUXAV/10.0[n]
. . F107 = OUTDATA(NOUT).FLUXPR/10.0[o]
. . For J = 1 to 7
. . . AP(J) = OUTDATA(NOUT).AP(J)/10.0
. . Next J
. . Invoke MSIS-90 routine GTD6(IYD, SEC, ALT, GLAT, GLONG, STL,
      F107A, F107, AP, D, T) to obtain mass and number densities
      (D) and local and exospheric temperatures (T);
. . [Calculate the mean molecular weight, in Atomic Mass Units,
      from the components]
. . WtMol = (D(6)/(D(1) + D(2) + D(3) + D(4) + D(5) + D(7) +
      D(8)))/AMU[p]
. . [Store values in output structure]
. . OUTDATA(NOUT).MDEN = Round(10^15×D(6))
. . OUTDATA(NOUT).MWT = Round(1000×WtMol)
. . OUTDATA(NOUT).MTEMP = Round(T(2))

. . {SET_SOLVE} [Set up variables for density and wind solution,
      according to option selected]
. . If CALOPT = 0 Then
. . . ThermT = T(2)[q]
. . Else If CALOPT = 10 Then
. . . WtMol = OUTDATA(NOUT).AWT/1000.0
. . . ThermT = T(2)
. . Else If CALOPT = 11 Then
. . . WtMol = OUTDATA(NOUT).AWT/1000.0
. . . ThermT = T(2)
. . . Dens = 10^-15×OUTDATA(NOUT).ADEN
. . Else If CALOPT = 20 Then
. . . WtMol = OUTDATA(NOUT).CWT/1000.0
. . . ThermT = T(2)
. . Else If CALOPT = 21 Then
. . . ThermT = OUTDATA(NOUT).CTEMP
. . Else If CALOPT = 22 Then
. . . WtMol = OUTDATA(NOUT).CWT/1000.0
. . . ThermT = OUTDATA(NOUT).CTEMP
. . Else If CALOPT = 23 Then
. . . WtMol = OUTDATA(NOUT).CWT/1000.0
. . . ThermT = OUTDATA(NOUT).CTEMP
. . . Dens = 10^-15×OUTDATA(NOUT).CDEN
. . Else If CALOPT = 30 Then
. . . Dens = 10^-15×OUTDATA(NOUT).MDEN
. . . ThermT = T(2)
. . Else
. . . [Report an invalid calculation option]
```

```
. . . Report CALOPT and allowable options to user and LOG;
. . . Proceed to {END_DENSITY};
. . End If
. . [Obtain the rotation rates in radians/sec, in satellite
      coordinates]
. . Rot1 = RadCnv*OUTDATA(NOUT).ROTR/3600.0[r]
. . Rot2 = RadCnv*OUTDATA(NOUT).ROTP/3600.0[s]
. . Rot3 = RadCnv*OUTDATA(NOUT).ROTY/3600.0[t]
. . RotSq = Rot1**2 + Rot2**2 + Rot3**2
. . [Determine the accelerometer offset from the center-of-mass,
      for satellite coordinates in meters]
. . Offset(1) = 0.001*(POS1 - OUTDATA(NOUT).CG1)
. . Offset(2) = 0.001*(POS2 - OUTDATA(NOUT).CG2)
. . Offset(3) = 0.001*(POS3 - OUTDATA(NOUT).CG3)
. . RotPrj = Rot1*OffSet(1) + Rot2*OffSet(2) + Rot3*OffSet(3)
. . [Determine the gravity gradient corrections, to first order,
      assuming the attitude angles are with respect to the Local
      Vertical/Ram Direction system]
. . OrbRad = OUTDATA(NOUT).RAD
. . Pitch = RadCnv*(OUTDATA(NOUT).PITCH/60.0)
. . Yaw = RadCnv*(OUTDATA(NOUT).YAW/60.0)
. . Roll = RadCnv*(OUTDATA(NOUT).ROLL/60.0)
. . Invoke EulTrns(Pitch,Yaw,Roll,XForm) to compute the
      transformation matrix for the specified attitude;
. . Invoke IMSL:MURRV(3,3,XForm,3,3,RadVec,1,3,VertRef) to
      transform (using matrix multiplication by the transpose of
      XForm) the local unit radius vector (RadVec) in the nominal
      satellite coordinate system (vertical and ram alignment) to
      a reference vertical (VertRef) in current satellite
      coordinates;
. . RadPrj = OffSet(1)*VertRef(1) + OffSet(2)*VertRef(2) +
      OffSet(3)*VertRef(3)
. . For K = 1 to 3
. . . GGrad(K) = GAcc*(REarth/OrbRad)**2*(OffSet(K) -
        3*RadPrj*VertRef(K))/OrbRad
. . Next K
. . [Calculate drag force (in newtons) in satellite coordinates,
      correcting for accelerometer biases, rotational effects,
      and gravity gradient]
. . Drag(1) = OUTDATA(NOUT).SMASS*
      (ASCALE*GConv*(OUTDATA(NOUT).ACCZ - OUTDATA(NOUT).BIASZ) +
      (RotPrj*Rot1 - RotSq*OffSet(1)) + GGrad(1))
. . Drag(2) = OUTDATA(NOUT).SMASS*
      (-ASCALE*GConv*(OUTDATA(NOUT).ACCX - OUTDATA(NOUT).BIASX) -
      (RotPrj*Rot2 - RotSq*OffSet(2)) + GGrad(2))
. . Drag(3) = OUTDATA(NOUT).SMASS*
      (-ASCALE*GConv*(OUTDATA(NOUT).ACCY - OUTDATA(NOUT).BIASY) -
      (RotPrj*Rot3 - RotSq*OffSet(3)) + GGrad(3))
. . [Calculate the expected bulk gas flow at the satellite, based
      on the satellite motion and the earth's rotation, but
```

```
    neglecting winds]
. . VTrnsv = Sqrt(OUTDATA(NOUT).VTHETA**2 +
    OUTDATA(NOUT).VPHI**2)[u]
. . VCoRot = OUTDATA(NOUT).RAD*Omega*
    CosD(OUTDATA(NOUT).LAT/100.0)[v]
. . ECVel(1) = VCoRot*OUTDATA(NOUT).VPHI/VTrnsv - VTrnsv
. . ECVel(2) = -VCoRot*OUTDATA(NOUT).VTHETA/VTrnsv
. . ECVel(3) = OUTDATA(NOUT).VRAD
. . Invoke IMSL:MURRV(3,3,XForm,3,3,ECVel,1,3,Vel) to transform
    (using matrix multiplication by the transpose of XForm) the
    local earth-centered velocity vector (ECVel) in the nominal
    satellite coordinate system (vertical and ram alignment) to
    a reference velocity (Vel) in current satellite
    coordinates;

. . {SEL_SOLVE} [Select solution method, based on use of
    available density data]
. . If CALOPT = 11 or CALOPT = 23 or CALOPT = 30 Then
. . . Invoke WINDS(WtMol, ThermT, Vel, Drag, Dens, WindVel,
      DrgCoef) to calculate the drag coefficients and wind
      velocity components in satellite coordinates based on the
      mean molecular weight, gas temperature, bulk gas
      velocity, drag force, and density estimate;
. . . [Assign an appropriate value for the zero-order
      accelerometer density]
. . . If InType = 'DENSITY ' Then
. . . . [Carry previous value to new output data]
. . . . OUTDATA(NOUT).DEN0 = SRC(I).DEN0
. . . . OUTDATA(NOUT).DEN = SRC(I).DEN
. . . Else
. . . . OUTDATA(NOUT).DEN0 = 0
. . . . OUTDATA(NOUT).DEN = 0
. . . End If
. . Else
. . . Invoke DENWND(WtMol, ThermT, Vel, Drag, Dens0, Dens,
      WindVel, DrgCoef) to calculate the drag coefficients,
      density, and wind velocity components in satellite
      coordinates based on the mean molecular weight, gas
      temperature, bulk gas velocity, and drag force;
. . . OUTDATA(NOUT).DEN0 = Round(10^15×Dens0)
. . . OUTDATA(NOUT).DEN = Round(10^15×Dens)
. . End If
. . [Store drag coefficient, in satellite coordinates, and wind
    results, in accelerometer coordinates]
. . OUTDATA(NOUT).CD1 = Round(1000*DrgCoef(1))
. . OUTDATA(NOUT).CD2 = Round(1000*DrgCoef(2))
. . OUTDATA(NOUT).CD3 = Round(1000*DrgCoef(3))
. . OUTDATA(NOUT).WINDX = Round(-WindVel(2))
. . OUTDATA(NOUT).WINDY = Round(-WindVel(3))
. . OUTDATA(NOUT).WINDZ = Round(WindVel(1))
```

```
. . [Write data block to DENSITY when full block is accumulated]
. . If NOUT ≥ MAXOUT Then
. . . Write NOUT, (OUTDATA(L), L = 1, NOUT) to DENSITY;
. . . NOUT = 0
. . End If
. Next I
. Proceed from {GET_DATA};

{END_DENSITY} [Write out partial block, and conclude processing]
. If NOUT > 0 Then
. . Write NOUT, (OUTDATA(L), L = 1, NOUT) to DENSITY;
. . Close DENSITY;
. End If
. Exit program;
```

Subroutines

NDAYS(IYR,IMON,IDAY)
[Calculate the SETA day number for specified calendar date]
IYR = (I*2) calendar date year
IMON = (I*2) calendar date month number
IDAY = (I*2) calendar date day of month
See Function definition in Raw Data Unpacking or Raw Data Checking programs

---

LINTRP(X,Y,X0)
[Perform linear interpolation over two (X,Y) pairs to obtain Y value at X0 (as REAL*8 value LINTRP)]
X(2) = (R*8) independent variable for interpolation
Y(2) = (R*8) dependent variable for interpolation
X0 = (R*8) selected value requiring dependent value

```
D1 = X0 - X(1)
D2 = X0 - X(2)

X12 = X(1) - X(2)

LINTRP = (Y(1)*D2 - Y(2)*D1)/X12

Return to calling routine;
```

GCINTS(T,LAT,LON,T0,LAT0,LON0)
[Perform interpolation along an arc for an intermediate position, for spherical coordinate inputs]
T(2) = (R*8) initial and final times, for limits of arc (as day and fraction of day)
LAT(2) = (R*8) initial and final "latitudinal" positions along arc (degrees)
LON(2) = (R*8) initial and final "longitudinal" position along arc (degrees)
T0 = (R*8) specified time for intermediate position
LAT0 = (R*8) intermediate "latitudinal" position at specified time (degrees)
LON0 = (R*8) intermediate "longitudinal" position at specified time (degrees)

```
For I = 1 to 2
. X(I) = CosD(LAT(I))*CosD(LON(I))
. Y(I) = CosD(LAT(I))*SinD(LON(I))
. Z(I) = SinD(LAT(I))
Next I

Invoke GCINTR(T,X,Y,Z,T0,LAT0,LON0) to interpolate the sampled
```

```
  rectangular coordinates X, Y, Z to calculate the spherical
  coordinate angles at time T0;
Return to calling routine;
```

```
GCINTR(T,X,Y,Z,T0,LAT0,LON0)
  [Perform interpolation along an arc for an intermediate
    position, for rectangular coordinate inputs]
  T(2) = (R*8) initial and final times, for limits of arc (as day
    and fraction of day)
  X(2) = (R*8) initial and final X-coordinate positions along arc
  Y(2) = (R*8) initial and final Y-coordinate position along arc
  Z(2) = (R*8) initial and final Z-coordinate position along arc
  T0 = (R*8) specified time for intermediate position
  LAT0 = (R*8) intermediate "latitudinal" position at specified
    time (degrees)
  LON0 = (R*8) intermediate "longitudinal" position at specified
    time (degrees)

[Calculate the full and partial time intervals]
DT21 = T(2) - T(1)
DT01 = T0 - T(1)

[Calculate the dot product and angular separation for the end
  points]
PROJ = X(1)*X(2) + Y(1)*Y(2) + Z(1)*Z(2)
OMEGA = ACOS(PROJ)

[Calculate the angular separation for the interpolated point, and
  the associated interpolation coefficients]
DELTA = (DT01/DT21)*OMEGA
A = Sin(DELTA)/Abs(Sin(OMEGA))
B = Cos(DELTA) - A*PROJ

[Calculate the rectangular coordinates of the interpolated point]
X0 = A*X(2) + B*X(1)
Y0 = A*Y(2) + B*Y(1)
Z0 = A*Z(2) + B*Z(1)

[Calculate the spherical angle coordinates of the interpolated
  point]
LAT0 = ASinD(Z0)
LON0 = ATan2D(Y0,X0)

Return to calling routine;
```

```
WINDS(WtMol, ThermT, Vel, Drag, Dens, WindVel, CD)

Input:
  WtMol = mean molecular weight, in atomic mass units;
  ThermT = ambient temperature, in degrees Kelvin;
  Vel(3) = zero-order ambient flow velocity components with
    respect to satellite, in m/sec;
  Drag(3) = drag force components on satellite, in newtons;
  Dens = ambient mass density, in g/cm^3;

Output:
  WindVel(3) = wind velocity components with respect to
    satellite, in m/sec;
  CD(3) = drag coefficient components (dimensionless);

"External": DragEq

[Set parameters for IMSL solution routine NEQNF]
  NDim = 3 [dimension of Vel, Drag, WindVel, and Flow]
  ErrRel = 0.01 [relative error targeted for successive
    approximate solutions]
  ItMax = 20 [maximum number of iterations]
```

Note: May want to control ErrRel and ItMax through user input to main routine.

```
[Initial net flow estimate uses wind result from previous sample]
For K = 1 to 3
. Flow(K) = Vel(K) + WindVel(K)
Next K

Invoke IMSL:NEQNF(DragEq, ErrRel, NDim, ItMax, Flow, WindVel,
  Discrep) to solve the drag equations for a net flow velocity
  WindVel;

[Extract the wind component from the net flow solution]
For K = 1 to 3
. WindVel(K) = WindVel(K) - Vel(K)
Next K

Return to calling routine;
```

```
DragEq(TotVel, DragDif, KEquat)

Input:
  TotVel(3) = vector of total velocity with respect to satellite,
    in m/sec;
  KEquat = Number of velocity components and drag equations (must
    be 3);
```

Output:
  DragDif(3) = vector of differences between calculated and measured drag;

Global input variables:
  Dens = ambient density, in g/cm^3;
  Drag(3) = measured drag, in newtons;
  WtMol = average molecular weight, in atomic mass units;
  ThermT = ambient temperature, in degrees Kelvin;
  CMass(3) = center-of-mass coordinates, in meters;
  ARef = frontal reference area, in square meters;

Global output variables:
  CD(3) = drag coefficient values for each axis;

Local variables:
  Torque(3) = torque values about each axis;
  VMagSq = square of total velocity magnitude;

```
[Calculate the drag coefficient components]
  Invoke DrgCoef(TotVel, WtMol, ThermT, CMass, CD, Torque)

[Calculate the drag differences, including conversion factor for
  Dens (g/cm^3 to kg/m^3)]
VMagSq = TotVel(1)**2 + TotVel(2)**2 + TotVel(3)**2
For I = 1 to 3
. DragDif(I) = 0.5*10^3*Dens*VMagSq*CD(I)*ARef - Drag(I)
Next I

Return to calling routine;
```

DENWND(WtMol, ThermT, Vel, Drag, Dens0, Dens, WindVel, CD)

Input:
  WtMol = mean molecular weight, in atomic mass units;
  ThermT = ambient temperature, in degrees Kelvin;
  Vel(3) = zero-order ambient flow velocity components with respect to satellite, in m/sec;
  Drag(3) = drag force components on satellite, in newtons;

Output:
  Dens0 = zero-order estimate for ambient mass density, in g/cm^3;
  Dens = final estimate for ambient mass density, in g/cm^3;
  WindVel(3) = wind velocity components with respect to satellite, in m/sec;
  CD(3) = drag coefficient components (dimensionless), as global output from WindEq;

SETA/ADS Software Development
Density and Winds Calculation Program

```
Global input variables:
  CMass(3) = center-of-mass coordinates coordinates for
    satellite, in meters;
  ARef = reference frontal area for satellite, in square meters;

"External": WindEq

Local:
  Est0(4) = initial solution estimate, as the set of values
    {Vel+WindVel, Dens0}
  Unknown(4) = computed solution, as the set of values
    {Vel+WindVel, Dens}
  WindMag = solution value for wind magnitude, in (m/sec)^2
  Torque(3) = torque components on satellite, in nt-m

[Set parameters for IMSL solution routine NCONF]
  NConstr = 3 [total number of constraints, inequality and
    equality (all equality)]
  NEqual = 3 [number of equality constraints]
  NVars = 4 [number of unknown variables]
  IBType = 0 [bounds supplied by user]
  UnkLBnd(4) = {-10^4, -10^4, -2x10^4, 0} [lower bounds for unknowns]
  UnkUBnd(4) = {10^4, 10^4, 2x10^4, 1.0} [upper bounds for unknowns]
  DiagScl(4) = {1.0, 1.0, 1.0, 1.0} [diagonal scale values for
    unknowns]
  IPrint = 0 [no printed output]
  ItMax = 20 [maximum number of iterations]
```

Note: May want to control IPrint and ItMax through user input to main routine.

```
[Initial net flow estimate uses wind result from previous sample,
  and current zero-order density estimate (without winds)]
Invoke DrgCoef(Vel, WtMol, ThermT, CMass, CD, Torque) to
  calculate the drag coefficient and torque components for the
  zero-order flow velocity;
VSq = Vel(1)**2 + Vel(2)**2 + Vel(3)**2
[Calculate zero-order density using in-track axis]
Dens0 = Drag(1)/(0.5*10^3*CD(1)*ARef*VSq)
Est0(4) = Dens0
For K = 1 to 3
. Est0(K) = Vel(K) + WindVel(K)
Next K

Invoke IMSL:NCONF(WindEq, NConstr, NEqual, NVars, Est0, IBType,
  UnkLBnd, UnkUBnd, DiagScl, IPrint, ItMax, Unknown, WindMag) to
  solve the augmented drag equations for a net flow velocity and
  density;
```

```
[Extract the wind component from the net flow solution]
For K = 1 to 3
. WindVel(K) = Unknown(K) - Vel(K)
Next K
Dens = Unknown(4)

Return to calling routine;
```

```
WindEq(NConstr, NEqual, NVars, Spec, Select, Result, Constr)

Input:
  NConstr = total number of constraints;
  NEqual = number of equality constraints;
  NVars = number of unknown variables;
  Spec(4) = current specification of the unknown variables {total
    velocity components, density};
  Select(4) = logical array for selecting active constraints;

Output:
  Result = value of augmented wind function (minimization
    condition);
  Constr(3) = values of the drag equation (constraint)
    differences;

Global input variables:
  Vel(3) = zero-order ambient flow velocity components with
    respect to satellite, in m/sec;
  Drag(3) = measured drag, in newtons;
  WtMol = average molecular weight, in atomic mass units;
  ThermT = ambient temperature, in degrees Kelvin;
  CMass(3) = center-of-mass coordinates for satellite, in meters;
  ARef = frontal reference area for satellite, in square meters;

Global output variables:
  CD(3) = final drag coefficient values for each axis;

Local variables:
  Dens = ambient density, in g/cm^3;
  TotVel(3) = vector of total velocity with respect to satellite,
    in m/sec;
  Torque(3) = torque values about each axis;
  VMagSq = square of total velocity magnitude;

[Calculate the drag coefficient components]
For I = 1 to 3
. TotVel(I) = Spec(I)
Next I
Invoke DrgCoef(TotVel, WtMol, ThermT, CMass, CD, Torque) to
  calculate the drag coefficient and torque components for the
```

```
  current total flow velocity;

[Calculate the drag differences, including conversion factor for
  Dens (g/cm^3 to kg/m^3)]
Dens = Spec(4)
VMagSq = TotVel(1)**2 + TotVel(2)**2 + TotVel(3)**2
For I = 1 to NEqual
. If Select(I) is TRUE Then Constr(I) =
    0.5*10^3*Dens*VMagSq*CD(I)*ARef - Drag(I)
Next I

[Calculate the minimization condition]
Result = (TotVel(1) - Vel(1))**2 + (TotVel(2) - Vel(2))**2 +
  (TotVel(3) - Vel(3))**2

Return to calling routine;
```

SETA/ADS Software Development
Density and Winds Calculation Program

Definitions and Notes

a. MASS = selection setting for molecular species in MSIS-90

b. NOUT = number of output samples for DENSITY block, and index of current sample

c. APDT = SETA day and fraction for center time of current 3-hour A_p value

d. WindVel = wind velocity components in satellite coordinates, in m/sec

e. IReq = selector for data type required (SETA, ADMS, CADS, MSIS) for processing option

f. MYr = 4-digit year for solar activity/flux data
Mon = 2-digit month for solar activity/flux data
MDay = 2-digit day-of-month for solar activity/flux data
Vals Structure:
APDaily = daily A_p value
AP(8) = successive 3-hour A_p values for the day
Flux = daily $F_{10.7}$ flux for the day
Flux90 = average $F_{10.7}$ flux over the previous three months (90 days)

g. AP(1) = daily A_p
AP(2) = 3-hour A_p for current time
AP(3) = 3-hour A_p for three hours before current time
AP(4) = 3-hour A_p for six hours before current time
AP(5) = 3-hour A_p for nine hours before current time
AP(6) = average of eight 3-hour A_p indices for 12 to 33 hours prior to current time
AP(7) = average of eight 3-hour A_p indices for 36 to 59 hours prior to current time

h. IYD = day-of-year, from 1 to 365; (need additional mod(IYD-1,365) + 1 to strictly enforce 365-day limit)

i. SEC = Universal Time in seconds

j. ALT = altitude in kilometers

k. GLAT = geocentric latitude in degrees (should be geodetic for MSIS-90)

l. GLONG = geocentric longitude in degrees, positive East (should be geodetic for MSIS-90)

m. STL = local apparent solar time in hours

n. F107A = three-month average of F10.7 flux

o. F107 = daily F10.7 flux for previous day

p. WtMol = mean molecular weight, in atomic mass units

q. ThermT = local ambient gas temperature, in degrees Kelvin

r. Rot1 = rotation rate about satellite X-axis, in radians/sec (Roll)

s. Rot2 = rotation rate about satellite Y-axis, in radians/sec (Pitch)

t. Rot3 = rotation rate about satellite Z-axis, in radians/sec (Yaw)

u. VTrnsv = total transverse (horizontal) satellite velocity

v. VCoRot = co-rotating thermospheric velocity at satellite altitude