AD613294

ESD-TDR-65-76

SPIRAL DECAY AND SENS[illegible] CALIBRATION DIFFERENTIAL CORRECT[illegible] PROGRAMS

Volume III. Programmer's Manual

TECHNICAL DOCUMENTARY REPORT NO. ESD-TDR-65-76

FEBRUARY 1965

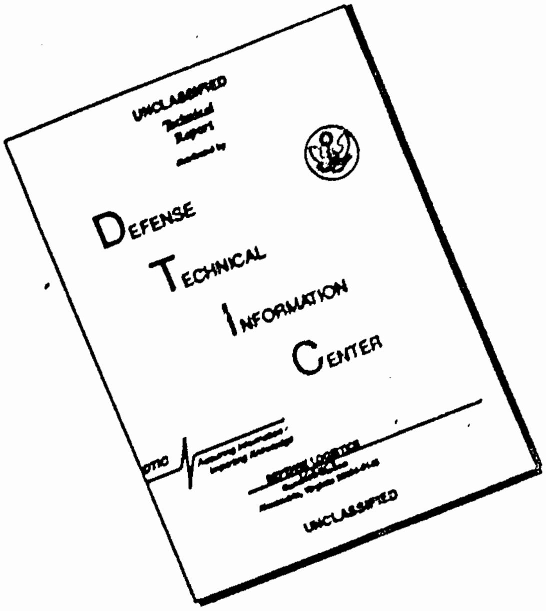
T. G. Gaa
C. G. Hilton
P. A. VanderStucken
L. G. Walters

2568

COPY 2 OF 2
HARD COPY $. 6.00
MICROFICHE $. 1.50

496L SYSTEM PROGRAM OFFICE
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

Prepared under Contract No. AF 19(628)-3377 by Aeronutronic, a Division of Philco Corporation, Newport Beach, California



UNCLASSIFIED
Technical
Report

DEFENSE
TECHNICAL
INFORMATION
CENTER

DTIC

UNCLASSIFIED

DDC AVAILABILITY NOTICE

ESD-TDR-65-76

SPIRAL DECAY AND SENSOR CALIBRATION DIFFERENTIAL CORRECTION PROGRAMS

Volume III. Programmer's Manual

TECHNICAL DOCUMENTARY REPORT NO. ESD-TDR-65-76

FEBRUARY 1965

T. G. Gaa
C. G. Hilton
P. A. VanderStucken
L. G. Walters

496L SYSTEM PROGRAM OFFICE
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

Prepared under Contract No. AF 19(628)-3377 by Aeronutronic, a Division of Philco Corporation, Newport Beach, California

FOREWORD

This Technical Documentary Report has been prepared in four volumes, as follows:

Volume	Title	Contractor's Publication Number
I	Program Development	U-3005
II	Operating Instructions	U-3006
III	Programmer's Manual	U-3007
IV	Operations Summary (U)	S-2990

Publication of this technical documentary report does not constitute Air Force approval of its findings or conclusions. It is published only for the exchange and stimulation of ideas.

CONTENTS

SECTION PAGE

1 INTRODUCTION 1

2 PROGRAM DESCRIPTION 2

2.1 Functional Description 2
2.2 Flow Diagrams. 4

3 MEMORY ASSIGNMENTS 7

3.1 Symbolic Names and Definitions 7
3.2 Data Block and Tape Formats 24
3.3 Memory Storage Diagram 64

4 SUBROUTINE DESCRIPTIONS. 65

4.1 Alphabetical Index to Subroutine Descriptions 65
4.2 Subroutine Descriptions 69

APPENDIX

I ADAMS-BASHFORTH INTEGRATION 231

I-1 R-K Procedure for Starting 231
I-2 A-B Sixth Order Integration 233
I-3 A-B Interval Control and Modification. . . . 235
I-4 Special Consideration for $Y_{1,n}$ 236
I-5 Flow Charts and Work Region Description . . 237
I-6 How to Use Adbash. 246
I-7 Derivative Subroutine 246
I-8 Initialization 247
I-9 Sample Flow Chart 249
I-10 To Use Runge-Kutta Integration Only. 250

ILLUSTRATIONS

FIGURE		PAGE
1	Spiral Decay Basic Functional Diagram	5
2	Spiral Decay Main Flow Diagram	6
3	Sensor Weight and Bias Tape Format	62
4	Binary Ephemeris Tape Format	63
5	Memory Storage Diagram	64
6	CNTRL Flow Diagram	103
7	JNDRAG Flow Diagram	148
8	PCONTRL Flow Diagram	178

APPENDIX

I-1	Adams Bashforth Flow Diagram	237
I-2	ADBASH Application	249

SECTION 1

INTRODUCTION

This document describes, from a programmer's view, the coding, function, and logic of the Spiral Decay computer program. The Spiral Decay program operates in the SPACETRACK B-3 Semi-Automatic Programming System; and will accept sensor observations, assign weights and biases to the data, perform a differential correction and predict past or future position and velocity.

Section 2 of this document contains a functional description and basic flow charts of the program. Section 3 defines symbolic names, data and tape formats, and memory assignments. A detailed description of each subroutine used by Spiral Decay is given in Section 4. The Appendix describes the Adams-Bashforth integration routine which is used by Spiral Decay for ephemeris integration.

SECTION 2

PROGRAM DESCRIPTION

2.1 FUNCTIONAL DESCRIPTION

The Spiral Decay Program contains three main functions. These are (1) OBSERVATION WEIGHTING, (2) DIFFERENTIAL CORRECTION, and (3) PREDICTION. This program is designed to operate in the Schedule Tape Mode under control of the B-3 Executive Program.

a. Observation Weighting

The Observation Weighting function of the program is to apply corrections (biases) to the observational data used for differential correction, and to apply statistical weights to the data used in the correction procedure.

The Spiral Decay Program must be supplied with weight and bias information for each observation. If none is available, that observation will be omitted from the differential correction.

Within Spiral Decay, there is room for weight and bias data for thirty sensors. There are approximately fifteen sets of sensor weighting data assembled within Spiral Decay. The weighting data contained in Spiral Decay may be changed and/or increased by introducing a "weight tape" on logical tape 7.

b. Differential Correction

The differential correction function accepts observational data for which weight and bias information is available, and performs a weighted differential correction on the orbit element set. Ephemeris computation is carried out in a special perturbations variation-of-parameters formulation. This formulation numerically integrates the perturbative accelerations influencing the parameters of an instantaneous two-body reference orbit.

The numerical integration is performed by a sixth-order Adams-Bashforth integration routine. This routine works strictly in a variable step size mode, based on error control parameters.

Perturbations handled in the program include: zonal,tesseral, and sectorial harmonics, atmospheric drag, and solar radiation pressure. These effects may be controlled to incorporate any combination in a particular case.

A correction can be obtained for any or all of the six orbital elements, and may also include the satellite mass. This latter parameter actually represents a correction to the ballistic parameter, $C_D A/m$, where C_D is the drag coefficient, A is the satellite cross-sectional area, (m^2) and m is the satellite mass (kg.).

The correction process will repeat up to twice the maximum number of iterations, as specified by the user, or until the root-mean-square (RMS) of the accepted residuals has converged. Accepted residuals are those that are not rejected on the basis of an absolute maximum or a relative check based on a multiple of the residual RMS. When the correction process has converged, the epoch may be updated to any revolution number or time.

c. Prediction

The prediction function of the program is used to obtain future or past position and velocity data from an input element set or a corrected element from the differential correction section.

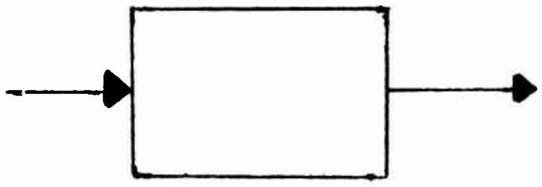
Ephemeris computation in the prediction function is carried out in the same manner as in the differential correction, with one addition: ephemeris integration, while a satellite is in the final decay stage, is performed using the Cowell equations. The Cowell integration uses a fourth-order Runge-Kutta integration scheme instead of Adams-Bashforth because of the need for frequent interval change.

In addition to a prediction ephemeris listed on hard copy, Spiral Decay can: (1) produce a binary ephemeris tape to be used with the XYZLA program; (2) leave in core five time points for use in a subsequent GIPAR run.

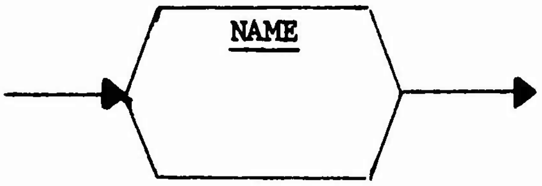
2.2 FLOW DIAGRAMS

The following two pages display the functional and subroutine level flow diagrams for Spiral Decay. Additional flow diagrams, written in further detail, are included in the individual subroutine descriptions in Section 4 and the Appendix of this manual.

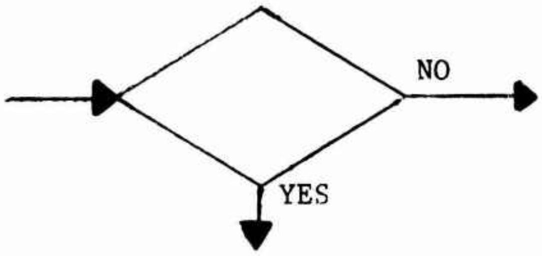
The following symbol conventions have been adopted:



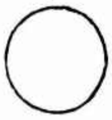
- contains processing function



- indicates a subroutine which is documented within the section or in a referenced system or publication



- indicates a decision or switch, including title on the appropriate path (may be more than two exits if a multiple switch)



- program or flow-diagram connector

- program terminal (start or exit)

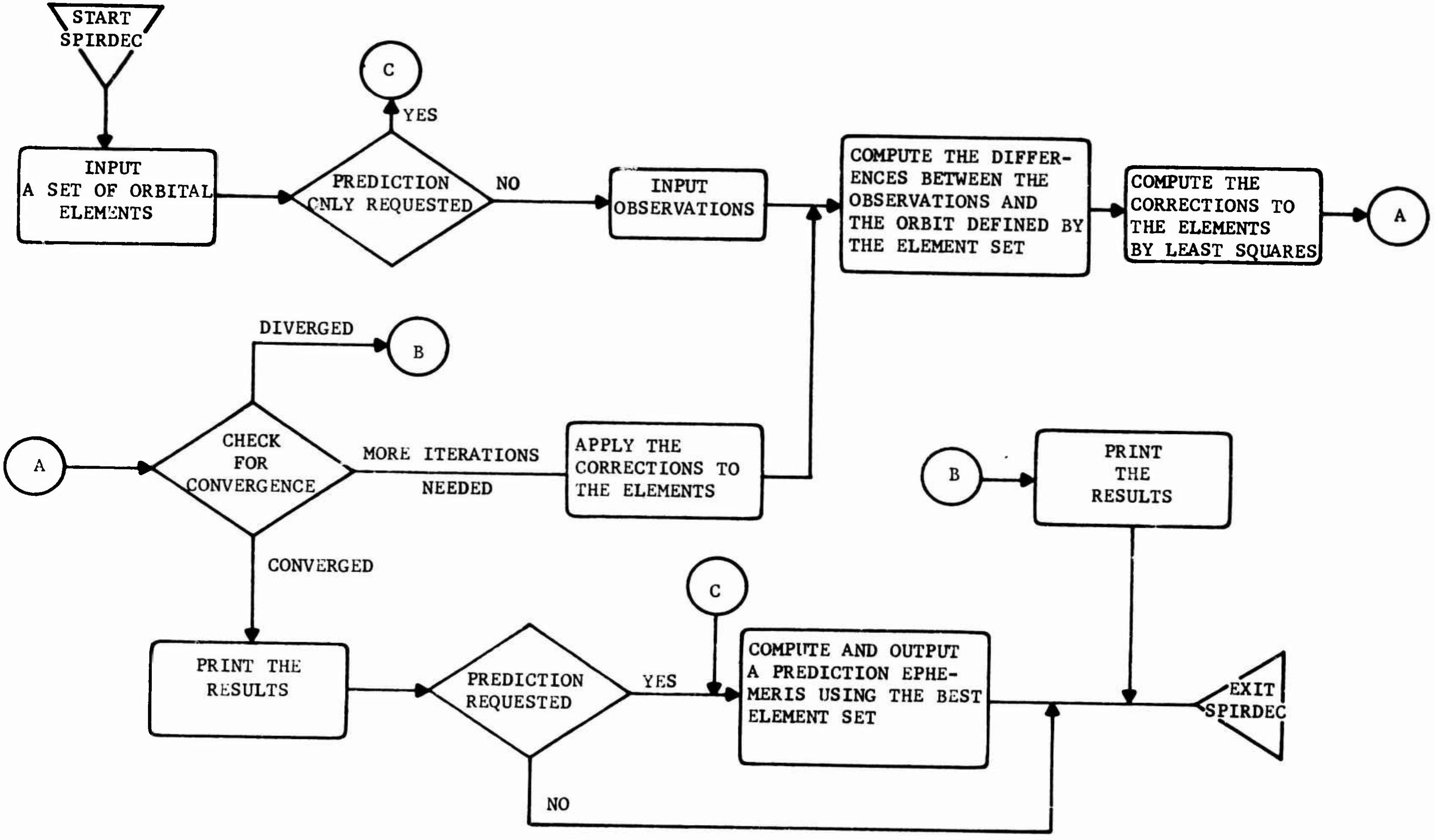


FIGURE 1. SPIRAL DECAY BASIC FUNCTIONAL DIAGRAM

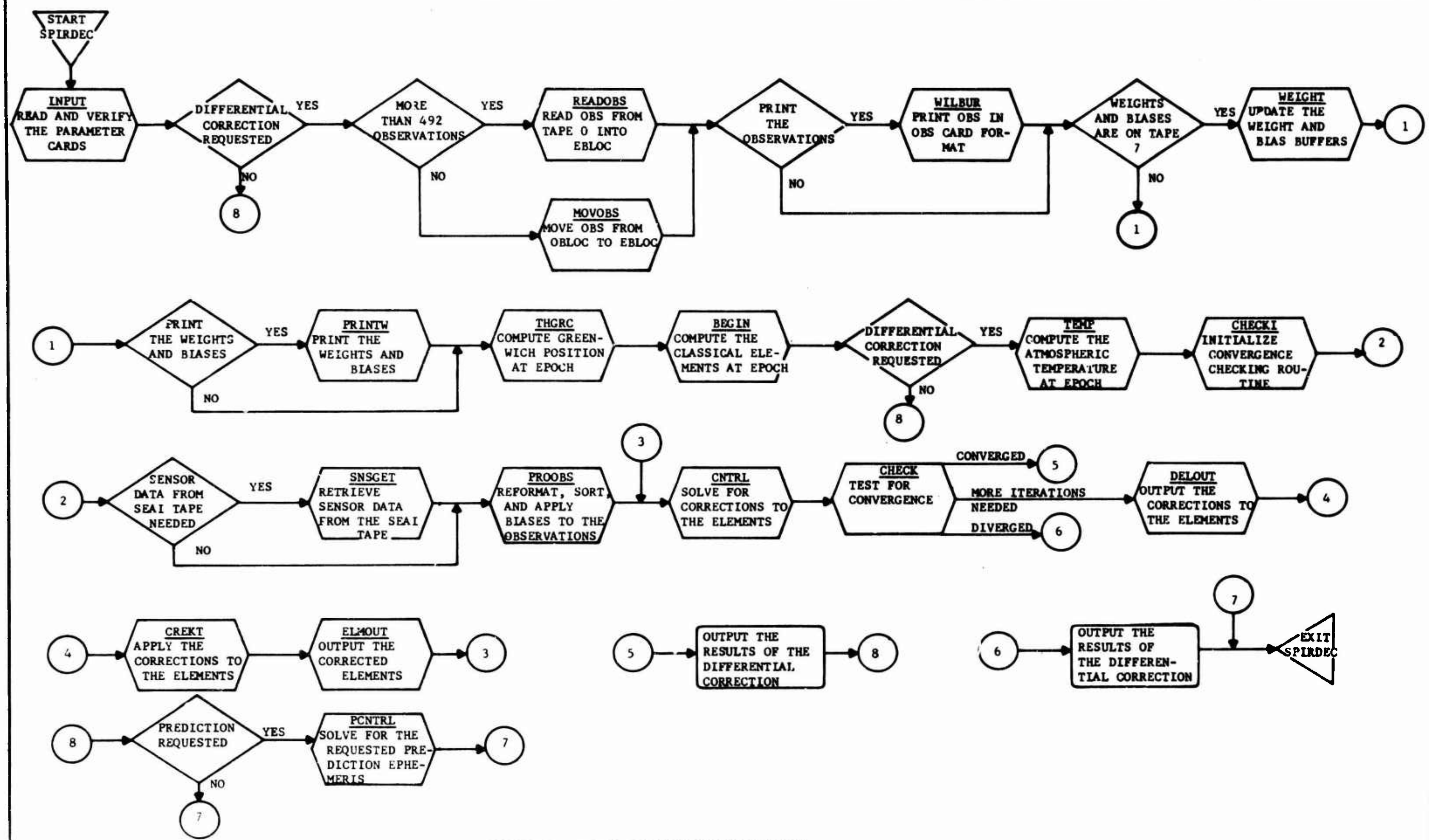


FIGURE 2. SPIRAL DECAY MAIN FLOW DIAGRAM

SECTION 3

MEMORY ASSIGNMENTS

3.1 SYMBOLIC NAMES AND DEFINITIONS

This section provides a listing and definitions of the symbolic names which have been created for the Spiral Decay Program, and are relative only to the coding of the program. The symbolic names which refer to address constants, input/output parameters, command constants, and temporary storage will not be listed in this section, but may be located directly in the code edit of Spiral Decay.

Definitions of some of the symbolic names contain mathematical symbols which are relative to the formulation of Spiral Decay. Definitions of these mathematical symbols may be found in Appendix VII, Volume 1.

The symbolic names which appear in the SPS Master Assign Deck may be referenced in either Aeronutronic report U-1691, Section 5-69, or System Development Corporation report TM-LX-38/000/00, Appendix E.

NAME	CELLS	DEFINITIONS
A11	49	Least Squares Matrix (N x N)
ABRHO	1	$(a \cdot B \cdot \rho_{\pi})/2$
ACCCNT	1	Accepted Residual Counter
ADDTO2	1	$\frac{1}{2}$ Integration Step Size
ADDTO3	1	$\frac{1}{3}$ Integration Step Size
ADLLEPS	1	Criteria for Increasing Integration Step Size
ADPERRK	1	Ratio of Runge Kutta to Adams Bashforth Integration Steps
ADP	1	Control Flag for Integration Routine
ADRKSTP	1	Indicator for Runge Kutta Integration
ADTEST	1	Integration Error
ADXR1	1	Index Register Storage
ADXR3	1	
ADXR5	1	
AE	1	a_e, Earth Radius = 1
ALFBUF	7	Buffer of Powers of α , $\alpha = \frac{-f(\sin^2 i)}{2h}\cos 2\omega$
ALFLG	1	α Residual Rejection Indicator
ALSUN	1	$\alpha_{\odot}$ = Right Ascension of Sun
AOVH	1	a/H
AP	1	Planetary Magnetic Index

NAME	CELLS	DEFINITIONS
ASCON2	1	6378165 = meters/e.r.
ASTK	1	One Word of ********
AXGR	1	a'_x
AXO	1	a_{x_o}
AXPRINT	1	a_x in Output Format
AX	1	a_x
AYGR	1	a'_y
AYO	1	a_{y_o}
AYPRINT	1	a_y in Output Format
AY	1	a_y
AZGR	1	a'_z
AZO	1	a_{z_o}
AZ	1	a_z
B11	7	Least Squares Matrix (1 x N)
BFLAG	1	Indicator for Inclusion of Bulge Perturbations
BIASAD	1	Address of Bias Buffer
BIBUF	181	Sensor Bias Buffer
BNADR	1	Address of BNBUF and BNNBUF
BNBUF	7	Buffer of (2n - 3) B_n, B_n = Bessel functions $n = 0, 1, \ldots 6$
BNNBUF	7	Buffer of (2n - 1) B_n, B_n = Bessel functions $n = 0, 1, \ldots 6$

NAME	CELLS	DEFINITIONS
BOO	56	Buffer Saving New Epoch Elements and ICK Buffer
BPRINT	1	B in Output Format
B	1	Ballistic Parameter
BTFLAG	1	Indicator for Binary Tape Output
BT	1	Output Time for Binary Tape
BUF	7	Temporary Buffer for Computing Correlation Matrix
CAPD	1	D
CAPR	1	R = Magnitude of Vector to Sensor
CAPX	1	X Location of Sensor
CAPY	1	Y
CAPZ	1	Z
CARDS	1	Indicator for "P Card" Input
CDLTAXN	1	$C\Delta a_{xn}$
CDLTAYN	1	$C\Delta a_{yn}$
CDLTN	1	$C\Delta n/_{n}$
CNFLAG	1	Indicator for n Only Correction on First Interation
COMPARE	1	Counter for Subroutine MARTINI
CON1	1	$-(\sin^2 i)f/2h$
CONBUF	128	Storage for "P Cards"
CONTEST	1	Criteria for D. C. Convergence

NAME	CELLS	DEFINITIONS
CORRDY	1	Temporaries used by CORRD subr.
CORRDZ	1	
COS2OM	1	Cosine 2 ω
COSEO	1	Cosine $e + \omega$
COSI	1	Cosine i
COSOM	1	Cosine ω
COSO	1	Cosine Ω
COSPH	1	Cos ϕ
COSPSI	1	Cos ψ
COSTH	1	Cosine θ
COSU	1	Cosine u
COUNTLR	1	Number of Elements to Correct - Left and Right Address
COUNTL	1	Number of Elements to Correct - Left Address
COUNTLX	1	Counter for MATRIX Subroutine
COUNTR	1	Number of Elements to Correct - Right Address
CRAIG	1	Indicator for Printed Observations
CSALS	1	Cos α_{θ}
CSNM	16	Tesseral Coefficient Buffer
DCFLAG	1	Indicator for a Differential Correction
DDGR	1	$\dot{D}$
DELTAQ	1	Δq
DENOM	1	$1 + \sqrt{1 - e^2}$

NAME	CELLS	DEFINITIONS
DFLAG	1	Indicator for Inclusion of Drag Perturbations
DGR	1	D'
DLFLG	1	δ Residual Rejection Indicator
DLSUN	1	$\delta_\odot$ = Declination of Sun
DLTB	1	$\Delta B/B$
DOTPRX	1	Temporaries for DOTPR Subroutine
DOTPRY	1	
DOTPRZ	1	
DQFLAG	1	Indicator for Delta Q Check
DQN	1	Maximum Delta Q
DRSDL	1	δ Residual
DUBMAT	36	Matrix for Increasing Integration Step Size
E2VGR	1	$-e^2 v'$
ECOSV	1	$p/r - 1$
ENDT	1	End Time for Printed Prediction Interval Output
E01	1	U = Used in Keplers Equation
EO	1	e_o
EPS1	1	Convergence Criteria for Iteration on Kepler's Equation
EPSILON	1	$\epsilon = 0.00001$
F10AV	1	Average Solar Radiation Constant
F10	1	Solar Radiation Constant

NAME	CELLS	DEFINITIONS
FIRST	1	Forward or Reverse Integration Indicator
FOBHR	1	Floating Point Hour, Min., Sec. in Output Format
FOBMIN	1	
FOBSEC	1	
FTFLAG	1	Indicator for Prediction by Revolution or Time
FX	1	Temporary Used by DIVDIF Subroutine
GAMMA	1	Reflectivity Constant
GE	1	g_e - With Zonal and Tesseral Effects
GEZ	1	g_e - With Only Zonal Effects
GITADR	1	Address of GIT Buffer
GIT	25	GIPAR Output Time Buffer
GS	1	g_s - With Zonal and Tesseral Effects
GSZ	1	g_s - With Only Zonal Effects
GU	1	g_u - With Zonal and Tesseral Effects
GUZ	1	g_u - With Only Zonal Effects
H1OVZ	1	250,000 Ft. in Earth Radii
H1	1	500,000 Ft. in Earth Radii
HAFMAT	36	Matrix for Decreasing Integration Step Size
HEAD5	6	Storage for BCD Page Heading
HEDLIN1	1	Parameters for Prediction Output Control
HEDLIN	1	
HSUBQP	1	Perigee Altitude (km) for Output

NAME	CELLS	DEFINITIONS
HSUBS	1	$1/H_\rho$
HXPRINT	1	h_x in Output Format
HYPRINT	1	h_y in Output Format
HZPRINT	1	h_z in Output Format
ICK	47	Interpolation Buffer
INELT	9	Initial Element Buffer
JBSCHNO	1	Input Option - From SPSJOB Card
JMPMOD	1	Jump Table Modifier in CDLTB Subroutine
JNSAVE	1	Save Index Registers
KAPPB	1	B Modifier (Altitude Dependent)
KAPPA	1	Upper Bound for B Modifier
KEORTM	1	.07436662
KNTRL	1	Indicator for Elements Being Corrected
LCOUNT	1	Counter for MARTINI Subroutine
LFLAG	1	Output Indicator for East or West Longitude
LINECNT	1	Current Line Position on Page
LINECT	1	Line Counter for RESOUT Subroutine
LJBUF	7	Temporary Storage for JNDRAG Subroutine
LJDC	1	$(\nu/2) \cdot B \cdot \rho_{\pi} \cdot {}_{B} \cdot 6378165$
LLO	1	ℓ_o
LNIOBE	1	$\frac{1}{10} \; LOG_e \; 10$
LOGRHOH	1	$\text{Log} \; \rho \; (h,T)$
LOLIMIT	1	Minimum Altitude Allowed for Drag Perturbations

NAME	CELLS	DEFINITIONS
LOLMT	1	Flag for Decay Corridor
LPREV	1	Temporary in LPMOD Subroutine
LPRINT	1	L in Output Format (deg.)
LPSUB	1	+ OR - 2π Used in LPMOD Subroutine
LPTEMP	1	Temporary in LPMOD Subroutine
LSQBUF	14	Buffer for Matrix Inversion
LSQX	7	Buffer of Delta Elements
LTBUF	6	"P Card 4" Parameter Storage
MARTSAV	1	Index Register Storage
MATRIXB	28	Buffer for Correlation Matrix
MAX	1	Maximum Acceptable Residual
MCOUNT	1	Counter for Matrix Subroutine
M	1	Counter in MARTINI Subroutine
MS2ERK	1	.00012649618 - Conversion from m/s to er/k
MU	1	$\mu = 1$
NEWPAGE	1	Line Counter for Page Control
NICOLET	200	Atmospheric Density Table
NPRFAC	12	Buffer of $1/(n + r)!$; $0 \leq (n + r) \leq 10$
N	1	Counter in MARTINI Subroutine
NU	1	ν
NUX	1	ν_x
NUY	1	ν_y

NAME	CELLS	DEFINITIONS
NUZ	1	ν_z
OANDE	1	Address of OBLOC and EBLOC
OBDAY	1	Day Number in BCD Output Format
OBFLG	1	Indicator of Observed Quantities
OBMO	1	Month Number in BCD Output Format
OBSLEFT	1	Counters for PROOBS Subroutine
OBSPROC	1	
OBSREJ	1	
OBYEAR	1	Year Number in BCD Output Format
OCOUNT	1	Temporary Counter
OLDRMS	1	RMS of Previous D.C.
OLDUZ	1	Previous Value of u_z
OLINCNT	1	Line Count of Printed Observations
OPRTESQ	1	$1 + \sqrt{1 - e^2}$
ORGDAY	1	Days from Beginning of Year to Epoch
ORMS	1	RMS of 1st Pass for Output
OVALA	1	Output Control for WILBUR Subroutine
P10LADR	1	10 Scaled T15
P1LADR	1	1 Scaled T15
P2LADR	1	2 Scaled T15
P3LADR	1	3 Scaled T15
P4LADR	1	4 Scaled T15

NAME	CELLS	DEFINITIONS
P55LADR	1	55 Scaled T15
P5LADR	1	5 Scaled T15
P6LADR	1	6 Scaled T15
P8LADR	1	8 Scaled T15
PAGENO	1	Output Page Counter
PAPRINT	1	Period - for Output
PAP	1	Planetary Magnetic Index - for Prediction
PARTA2	1	Temporaries for CDLTB Subroutine - Contain Segments of $\delta a/ \delta t$ and $\delta e/ \delta t$
PARTA3	1	
PARTA	1	
PARTE2	1	
PARTE3	1	
PARTE	1	
PB	1	Ballistic Parameter - for Prediction
PBUF	6	Buffer of P_n, n = 0, 1,..., 5
PDAY	1	BCD Day
PFIOAV	1	Average Solar Radiation Constant for Prediction
PF10	1	Solar Radiation Constant for Prediction
PFLAG	1	Indicator for Prediction Output
PGAMMA	1	Reflectivity Constant for Prediction
PHI	1	ϕ
PKAPPA	1	Lower Bound for B Modifier - for Prediction

NAME	CELLS	DEFINITIONS
POBUF	27	Buffer for Output of Observations
POVH	1	p/h
PPBUF	6	Buffer of P_n', n = 1,..., 5
PPPFLAG	1	Indicator for a Prediction
PRDSAV	1	Indicator for New Epoch Elements
PREDBF	11	Temporary Buffer
PREDFLG	1	Indicator for Prediction by Time or Revolution Number
PREV	1	Revolution Number for Prediction Output
PRTADR	1	Address of PRT Buffer
PRTIME	2	Revolution Number or BCD Date of New Epoch (D.C)
PRTIM	2	Revolution Number or BCD Time of Final Epoch (Prediction)
PRT	19	Buffer of Prediction Intervals for Output
PTCOUNT	1	Counter of Interpolation Points
PW	1	Indicator to Print Weights and Biases
PYEAR	1	BCD Year
QQ	8	Temporary Storage
RANGE	1	ρ
RBUF	9	Buffer of $R_{n,m}$
RCNT	1	Accepted Residual Counter
RDOTOR	1	$\dot{r}/r$
REJCNT	1	Rejected Residual Counter

NAME	CELLS	DEFINITIONS
REJFLG	1	Rejection Indicator for any Residual
RESCNTL	1	Control for Residual Output
RESOPT	1	Indicator for Type of Units of Printed Angle Residuals
REV	1	Revolution Number
RGFLG	1	Rejection Indicator for Range Residuals
RGSDL	1	Range Residual
RHO	1	Atmospheric Density = ρ
RKINDT	1	Initial Integration Step Size
RMS	1	Root Mean Square of Residuals
ROFLAG1	1	Residual Output Flags
ROFLAG	1	
ROVA	1	r/a
RPBUF	8	Buffer of $R_{n,m}'$
RPCON3	1	Radiation Pressure Constant
RPFLAG	1	Indicator for Inclusion of Radiation Pressure Perturbations
RPT	1	Maximum Number of Iterations for a Differential Correction
RRFLG	1	Rejection Indicator for Range Rate Residuals
RRSDL	1	Range Rate Residual
RSKNTRL	1	Indicator for Elements to be Corrected
RTIMUZ2	1	$\sqrt{1 - u_z^2}$ Used in MARTINI Subroutine

NAME	CELLS	DEFINITIONS
SATEL	1	Satellite Number (Left Justified)
SAVEO	1	Cell 0 Storage
SAVE3	1	Cell 3 Storage
SAVELEM	1	Indicator for Integrating to New Epoch
SAVEM	1	Temporary
SAVOBS	1	Temporary
SAVT	1	Temporary
SBUF	180	Buffer of Sensor Information
SCONBUF	70	Storage for "P Card" Images
SCOUNTR	1	Counter for MATRIX Subroutine
SIGMA1	1	$1/\sigma_{\rho}$ $(\text{e.r.})^{-1}$
SIGMA2	1	$1/\sigma_{\dot{\rho}}$ $(\text{e.r./ke})^{-1}$
SIGMA3	1	$1/\sigma_{A}$ $(\text{rad})^{-1}$
SIGMA4	1	$1/\sigma_{h}$ $(\text{rad})^{-1}$
SIGMAI	1	$1/\sigma_{i}$, σ_{i} = Weight for ρ, A, h, or $\dot{\rho}$
SIGN	7	Output Buffer for Standard Deviations
SIN2OM	1	Sine 2ω
SINBOTH	1	Sin $(\Psi + \xi)$
SINCOS	9	Buffer of sin m λ and cos m λ; m = 1,2,3,4
SKNTRL		Indicator for Elements to be Corrected
SMLGR	1	ℓ'
SNALS	1	Sin $\alpha_{\odot}$

NAME	CELLS	DEFINITIONS
SNDLS	1	Sin $\delta_\odot$
SOBFLG	1	OBLFG Storage
SOLDUZ	1	OLDUZ Storage
SREVF	1	Final Revolution Number to End Prediction
SREV	1	REV Storage
SUNLX	1	$L_{x_\odot}$
SUNLY	1	$L_{y_\odot}$
SUNLZ	1	$L_{z_\odot}$
TAPBUF	128	Buffer for Ephemeris Tape Output
TAPCNT	1	Current Address in TAPBUF
TEMPO	1	Atmospheric Temperature at Epoch
TEMP1	1	Temporary
TEMP2	1	Temporary
TEMP3	1	Temporary
TEMPT	1	Atmospheric Temperature at Any Time
TERMS	8	Buffer of 7 Coefficients and 1 Residual
TESRAL	1	Indicator for Inclusion of Tesseral Harmonics
THETA	1	θ
VALLINI	1	Control for Prediction Output
VALLIN	1	

NAME	CELLS	DEFINITIONS
WANDBI	1	Address of WBUF and BIBUF
WBUF	151	Buffer of Weights
WOBMARK	1	Temporary
XBDGR	1	$\dot{x}'_B$
XDD	1	$\ddot{x}$
XDGR	1	$\dot{x}'$
XDTGR	1	$\dot{x}'_D$
XKE	1	$k_e = .07436662$
XLAMD	1	λ for Output (deg)
XLSUNT	1	$L_\odot$ at Time t
XMPER	1	m/e. r. = 6378165
XRDGR	1	$\dot{x}'_R$
XRMODA	1	Counters for MATRIX Subroutine
XRMOD	1	
YBDGR	1	$\dot{y}'_B$
YDD	1	$\ddot{y}$
YDGR	1	$\dot{y}'$
YDTGR	1	$\dot{y}'_D$
YRDGR	1	$\dot{y}'_R$
YY	2	BCD Epoch Date
Z2BUF	7	Buffer of Powers of z/2, where $z = \frac{ae}{H}$

NAME	CELLS	DEFINITIONS
ZBDGR	1	$\dot{z}'_B$
ZDD	1	$\ddot{z}$
ZDGR	1	$\dot{z}'$
ZDTGR	1	$\dot{z}'_D$
ZRDGR	1	$\dot{z}'_R$
ZZ	1	$z = ae/H_\rho$

3.2 DATA BLOCK AND TAPE FORMATS

This section contains a definition of each of the buffers and tapes used by the Spiral Decay Program which are not common to the B-3 operating system.

Mathematical symbols used in the buffer descriptions are defined in Appendix VII, Volume I.

Buffer Name	Related Subroutines	Page
A11	LSQ, LSQS, LSQR	26
ALFBUF	CDLTB	27
B11	LSQ, LSQS, LSQR	28
BIFUF	WEIGHT, BIAS	29
BNBUF	CDLTBIN, CDLTB	30
BNNBUF	CDLTBIN, CDLTB	31
BOO	SAVICK, CNTRL, PCONTRL	32
BUF	MATRIX	33
CSNM	MARTINI	34
DUBMAT	ADBASH	35
EBLOC	PROOBS, NXTOB	36
GIT	PCONTRL	37
HAFMAT	ADBASH	38
ICK	DIVDIF, CNTRL, PCONTRL	39
JBUF	MARTINI	40
LJBUF	JNDRAG	41
LOGRHO	JNDRAG	42
LSQBUF	LSQS	43
LTBUF	PCONTRL	44
MATRIXB	MATRIX	45
NICOLET	JNDRAG	46
PBUF	MARTINI	47
POBUF	WILBUR	48
PPBUF	MARTINI	49
PREDBF	PCONTRL, CNTRL	50

Buffer Name	Related Subroutines	Page
RBUF	MARTINI	51
RPBUF	MARTINI	52
SBUF	SETSBUF, NXTOB	53
SIGN	MATRIX	54
SINCOS	MARTINI	55
TAPBUF	TAPEW	56
TBUF	DIVDIF	57
TERMS	CDLTB, LSQR, CMPCF	58
W	ADBASH	59
WBUF	WEIGHT, GETWGT	60
Z2BUF	CDLTBIN	61

Tape Number	Contents	Page
7	Sensor Weights and Biases	62
12	Binary Ephemeris for XYZLA	63

A11 49 Cells

A11 contains the least squares matrix used to solve for the corrections to the elements. For each accepted observation, a set of scalar differential coefficients are computed (C_i, i = 1...n; where n = no. of elements being corrected). Products of these coefficients are summed for each observation, and the result is the A11 matrix.

$$A11 = \begin{bmatrix} \Sigma C_1 C_1 & \Sigma C_1 C_2 & \cdot & \cdot & \cdot & \Sigma C_1 C_n \\ \Sigma C_2 C_1 & \Sigma C_2 C_2 & \cdot & \cdot & \cdot & \Sigma C_2 C_n \\ \cdot & \cdot & \cdot & & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & & \cdot & \cdot \\ \Sigma C_n C_1 & \Sigma C_n C_2 & \cdot & \cdot & \cdot & \Sigma C_n C_n \end{bmatrix}$$

The size of the A11 matrix depends on the number of elements being corrected. The matrix is stored sequentially by rows, where the rows and columns represent corrections to n, a_{xn}, a_{yn}, U_o, Ω, i, B, in that order. If only U_o and Ω were being corrected, A11 would be:

A11 + 0 $\Sigma C_{U_o} C_{U_o}$

+ 1 $\Sigma C_{U_o} C_{\Omega}$

+ 2 $\Sigma C_{\Omega} C_{U_o}$

+ 3 $\Sigma C_{\Omega} C_{\Omega}$

ALFBUF 7 Cells

ALFBUF is used in the CDLTB subroutine and contains powers of $(\alpha^n)/n!$ for n = 0, 1, 2, 3, 4, 5, 6.

ALFBUF	+ 0	$\alpha^0/0! = 1$
	+ 1	$\alpha^1/1!$
	+ 2	$\alpha^2/2!$
	+ 3	$\alpha^3/3!$
	+ 4	$\alpha^4/4!$
	+ 5	$\alpha^5/5!$
	+ 6	$\alpha^6/6!$

B11 7 Cells

B11 contains the least squares vector used to solve for the corrections to the elements. For each accepted observation, a set of scalar differential coefficients (C) and a residual (R) are computed. The products, C·R, are summed for each observation, and the result is the B11 matrix.

B11	+ 0	$\Sigma\, C\Delta n/_{n} \cdot R$
	+ 1	$\Sigma\, Ca_{xn} \cdot R$
	+ 2	$\Sigma\, Ca_{yn} \cdot R$
	+ 3	$\Sigma\, CU_{o} \cdot R$
	+ 4	$\Sigma\, C\Omega \cdot R$
	+ 5	$\Sigma\, C_{i} \cdot R$
	+ 6	$\Sigma\, C\Delta B/_{B} \cdot R$

B11 would appear as above if all seven elements were being corrected. B11 is actually compacted to represent only those elements being corrected. If only U_o and i were being corrected, B11 would be:

B11	+ 0	$\Sigma\, CU_{o} \cdot R$
	+ 1	$\Sigma\, C_{i} \cdot R$

BIBUF 181 Cells

BIBUF contains sensor bias information for range, azimuth, elevation, range rate and time. There are about 16 sets of sensor biases assembled in the program; these may be changed or extended by introducing weight and bias cards on logical tape 7.

BIBUF + 0 00000SSS (BCD Sensor Number[*])

+ 1 $-B_r$ (e.r.)

+ 2 $-B_a$ (e.r.)

+ 3 $-B_n$ (e.r.)

+ 4 $-B_{\dot{r}}$ (e.r./ke)

+ 5 $-B_T$ (min.)

.
.
.

Up to 29 sets of sensor number, B_r, B_a, B_h, $B_{\dot{r}}$, B_T, followed by 1 BCD word of 00000ZZZ

+ 180

---

[*] Bit position 0 of each word containing sensor number is set to 1 for each VERLORT/PRELORT sensor.

BNBUF 7 Cells

BNBUF contains factors of $B_n(z)$ which are computed in subroutine CDLTBIN and used in subroutine CDLTB.

BNBUF	+ 0	$B_0\ (z)$
	+ 1	$B_1\ (z)$
	+ 2	$B_2\ (z)$
	+ 3	$3 \cdot B_3\ (z)$
	+ 4	$3 \cdot 5 \cdot B_4\ (z)$
	+ 5	$3 \cdot 5 \cdot 7 \cdot B_5\ (z)$
	+ 6	$3 \cdot 5 \cdot 7 \cdot 9 \cdot B_6\ (z)$

BNNBUF 7 Cells

BNNBUF contains factors of B_n (z) which are computed in subroutine CDLTBIN and used in subroutine CDLTB.

BNNBUF	+ 0	B_0 (z)
	+ 1	B_1 (z)
	+ 2	$3 \cdot B_2$ (z)
	+ 3	$3 \cdot 5 \cdot B_3$ (z)
	+ 4	$3 \cdot 5 \cdot 7 \cdot B_4$ (z)
	+ 5	$3 \cdot 5 \cdot 7 \cdot 9 \cdot B_5$ (z)
	+ 6	$3 \cdot 5 \cdot 7 \cdot 9 \cdot 11 \cdot B_6$ (z)

B00 56 Cells

B00 is used to store the interpolation buffer (ICK) and new epoch elements as found during a D.C. This is done to save unnecessary integration when starting a prediction.

B00 + 0	a_{xn}	new epoch elements
+ 1	a_{yn}	
+ 2	B	
+ 3	a	
+ 4	e^2	
+ 5	sin i	
+ 6	p	
+ 7	n	
+ 8	time (min)	
+ 9	L	
+ 10	a_x	
+ 11	a_y	
+ 12	a_z	
+ 13	h_x	
+ 14	h_y	
+ 15	h_z	
+ 16 to + 55	5 sets of t, L, $\underline{a}$, $\underline{h}$, taken directly from ICK+0 to ICK+39	

BUF 7 Cells

BUF contains the standard deviations of the delta elements for those elements being corrected.

If seven elements were being corrected:

BUF + 0 $\sigma\ n$
\+ 1 $\sigma\ a_{xn}$
\+ 2 $\sigma\ a_{yn}$
\+ 3 $\sigma\ U_o$
\+ 4 $\sigma\ \Omega$
\+ 5 $\sigma\ i$
\+ 6 $\sigma\ B$

If only a_{xn}, Ω, B were being corrected:

BUF + 0 $\sigma\ a_{xn}$
\+ 1 $\sigma\ \Omega$
\+ 2 $\sigma\ B$

CSNM 16 Cells

CSNM contains the tesseral coefficients as input from P cards 7 and 8.

CSNM + 0	C_{22}
+ 1	S_{22}
+ 2	C_{31}
+ 3	S_{31}
+ 4	C_{32}
+ 5	S_{32}
+ 6	C_{33}
+ 7	S_{33}
+ 8	C_{41}
+ 9	S_{41}
+ 10	C_{42}
+ 11	S_{42}
+ 12	C_{43}
+ 13	S_{43}
+ 14	C_{44}
+ 15	S_{44}

DUBMAT 36 Cells

DUBMAT contains a 6 X 6 matrix used to change the difference table in subroutine ADBASH to represent a step size twice as large as the previous step size.

This matrix is stored by rows starting with the first element.

$$\text{DUBMAT} = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 4 & -4 & 1 & 0 & 0 \\ 0 & 0 & 8 & -12 & 6 & -1 \\ 0 & 0 & 0 & 16 & -32 & 24 \\ 0 & 0 & 0 & 0 & 32 & -80 \\ 0 & 0 & 0 & 0 & 0 & 64 \end{bmatrix}$$

EBLOC 11834 Cells

EBLOC is used for observation storage. The original 10 words per observation is cut to 6 words per observation by subroutine PROOBS. Up to 984 observations are stored chronologically in EBLOC as follows:

EBLOC + 0 Station I.D. (BCD, right adjusted)

+ 1 time (minutes since epoch)

+ 2 ρ (range in e.r.)

+ 3 $\dot{\rho}$ (range rate in e.r./ke)

+ 4 α azimuth or rt. ascen. (rad.)

+ 5 δ elevation or declin. (rad.)

$\left.\begin{array}{c}\cdot\\\cdot\\\cdot\\\cdot\\\cdot\\\cdot\\\cdot\end{array}\right\}$ Up to 983 sets of Sta. I.D., t, ρ, $\dot{\rho}$, α, δ followed by one word of 00000ZZZ

GIT 11 Cells

GIT contains the output times requested on "P card 10" to be left in core for a subsequent GIPAR run.

GIT + 0	One bit for each GIPAR time requested, right adjusted to T47; i.e., $\overline{0 \quad 0}$ 1111 for four output times.	
GIT + 1	t_1	
+ 2	t_2	Up to five points (minutes from prediction epoch); terminated by a word of $\overline{0 \quad 0}$
+ 3	t_3	
+ 4	t_4	
+ 5	t_5	

During initialization, all 11 cells of GIT are used for input and conversion of time from date to minutes from epoch.

HAFMAT 36 Cells

HAFMAT contains a 6 X 6 matrix used to change the difference table in subroutine ADBASH to represent a step size half as large as the previous step size.

This matrix is stored by rows starting with the first element.

$$\text{HAFMAT} = \begin{bmatrix} .5 & .125 & .0625 & .0390625 & .02734375 & .0205078125 \\ 0 & .25 & .125 & .078125 & .0546875 & .041015625 \\ 0 & 0 & .125 & .09375 & .0703125 & .0546875 \\ 0 & 0 & 0 & .0625 & .0625 & .0546875 \\ 0 & 0 & 0 & 0 & .03125 & .0390625 \\ 0 & 0 & 0 & 0 & 0 & .015625 \end{bmatrix}$$

ICK 47 Cells

ICK is used as a buffer containing the last five sets of elements produced by the integration routine. When an output time falls within the time span covered by the five points in the ICK buffer, a set of elements for the output time are obtained by interpolation and stored in the last 7 cells of ICK.

ICK + 0	Time	most recent point obtained by the integration routine
+ 1	L	
+ 2	a_x	
+ 3	a_y	
+ 4	a_z	
+ 5	h_x	
+ 6	h_y	
+ 7	h_z	
+ 8		4 more sets of T, L, $\underline{a}$, $\underline{h}$ in the order they were obtained
.		
.		
.		
+ 39		
+ 40	L	obtained by interpolation for any output time T.
+ 41	a_x	
+ 42	a_y	
+ 43	a_z	
+ 44	h_x	
+ 45	h_y	
+ 46	h_z	

JBUF 4 Cells

JBUF contains zonal coefficients either as assembled in the program or as input from P card 6.

JBUF	+ 0	J_2
	+ 1	J_3
	+ 2	J_4
	+ 3	J_5

LJBUF 7 Cells

LJBUF is used in the JNDRAG subroutine as a set of temporaries.

LJBUF	+ 0	$\sin^{2.5} \xi/2$	used for temperature computations
	+ 1	$\cos^{2.5} \eta/2$	
	+ 2	$\cos^{2.5} \tau/2$	
	+ 3		used as temporaries for interpolation to find density coefficients
	+ 4		
	+ 5		
	+ 6		

LOGRHO   16 Cells

LOGRHO contains four sets of atmospheric density coefficients extracted directly from the NICOLET table. The coefficients inserted in LOGRHO by subroutine JNDRAG are those which correspond to altitudes surrounding one point for which the density coefficients will be interpolated.

LOGRHO + 0	log ρ	-1
+ 1	A	
+ 2	B	
+ 3	C	
+ 4	log ρ	0
+ 5	A	
+ 6	B	
+ 7	C	
+ 8	log ρ	+1
+ 9	A	
+ 10	B	
+ 11	C	
+ 12	log ρ	+2
+ 13	A	
+ 14	B	
+ 15	C	

← point to be interpolated

↓ increasing altitude

LSQBUF   14 Cells

LSQBUF is a buffer for temporary storage as required by Philco subroutine FMAIN. LSQBUF is used in subroutine LSQS.

LTBUF 6 Cells

LTBUF contains prediction controls as extracted from P card 4.

LTBUF	+ 0	not used
	+ 1	prediction output interval (floating point)
	+ 2	bulge perturbation flag (T47)
	+ 3	drag perturbation flag (T47)
	+ 4	radiation pressure flag (T47)
	+ 5	prediction output interval for a binary ephemeris tape (floating point)

MATRIXB 28 Cells

MATRIXB contains the correlation matrix of the delta elements, for output purposes only. MATRIXB is computed by subroutine MATRIX and is a lower half matrix compacted to represent the number of elements being corrected. The correlation half matrix is stored in MATRIXB by rows for the elements being corrected in the following order n, a_{xn}, a_{yn}, U_o, Ω, i, B.

If n, U_o, Ω, and i were being corrected MATRIXB would be:

MATRIXB	+ 0	$\Gamma_{n,n} = 1$
	+ 1	$\Gamma_{U_{o,n}}$
	+ 2	$\Gamma_{U_o, U_o} = 1$
	+ 3	$\Gamma_{\Omega, n}$
	+ 4	Γ_{Ω, U_o}
	+ 5	$\Gamma_{\Omega, \Omega} = 1$
	+ 6	$\Gamma_{i,n}$
	+ 7	Γ_{i, U_o}
	+ 8	$\Gamma_{i, \Omega}$
	+ 9	$\Gamma_{i, i} = 1$

In printed form this would be:

$$\begin{array}{llll}
\Gamma_{n,n} & & & \\
\Gamma_{U_o,n} & \Gamma_{U_o, U_o} & & \\
\Gamma_{\Omega,n} & \Gamma_{\Omega, U_o} & \Gamma_{\Omega, \Omega} & \\
\Gamma_{i,n} & \Gamma_{i,U_o} & \Gamma_{i,\Omega} & \Gamma_{i,i}
\end{array}$$

NICOLET 202 Cells

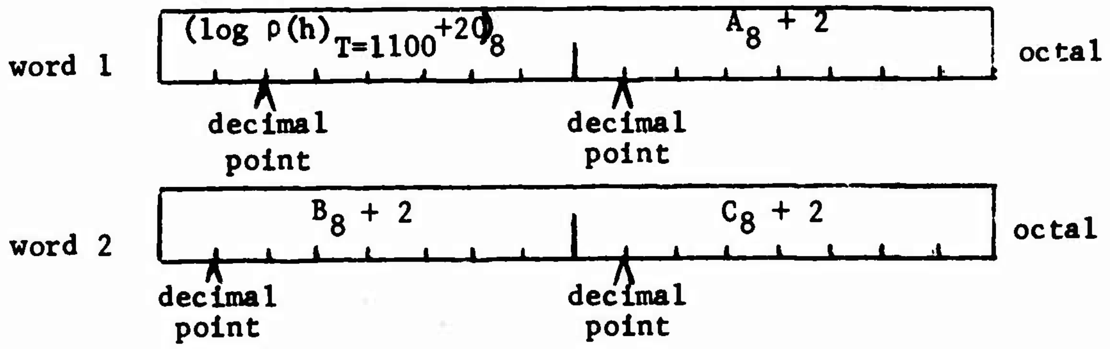
NICOLET is a table of atmospheric density coefficients vs. altitude. The table contains a set of coefficients for each ten kilometers in altitude from 0 to 1000 kilometers. The equation using these coefficients is:

$$\log \rho\ (h) = \log \rho\ (h)_{T=1100} + A\ \left(\frac{T-1100}{400}\right) + B\ \left(\frac{T-1100}{400}\right)^2 + C\ \left(\frac{T-1100}{400}\right)^3$$

where T = Temperature at h

h = altitude

Each entry in the table consists of two compacted octal words containing:



NICOLET	+ 0	word 1	at h = 0 km.
	+ 1	word 2	
	+ 2	word 1	at h = 10 km.
	+ 3	word 2	
	⋮		
	+ 200	word 1	at h = 1000 km.
	+ 201	word 2	

PBUF 6 Cells

PBUF is used in subroutine MARTINI for use in the computation of zonal harmonics. PBUF contains values of P_n for n = 0, 1, 2, 3, 4 and 5.

PBUF + 0 $P_0 = 1$
+ 1 $P_1 = U_z$
+ 2 P_2
+ 3 P_3
+ 4 P_4
+ 5 P_5

POBUF 27 Cells

POBUF is used by subroutine WILBUR, and contains the data from each observation card (one at a time) so that a hard copy listing of the input observations may be obtained.

POBUF +	0	Satellite No.	BCD
	1	Equipment type	BCD
	2	Station ID	BCD
	3	Accuracy	BCD
	4	YYMMDD (year,mo.,day)	BCD
	5	Hour	Fl. Pt.
	6	Minutes	Fl. Pt.
	7	Seconds	Fl. Pt.
	8	Elevation or declination	Fl. Pt.
	9	Range	Fl. Pt.
	10	Range rate	Fl. Pt.
	11	Max. frequency shift	BCD
	12	Brightness	BCD
	13	Maximum	BCD
	14	Minimum	BCD
	15	Time interval	BCD
	16	Not used	
	17	Message No.	BCD
	18	Equinox	BCD
	19	Not used	
	20	Observation number	BCD
	21	Not used	
	22	Right ascension indicator	BCD
	23	Azimuth or Rt.ascen.seconds	Fl. Pt.
	24	Rt. Ascen. minutes	Fl. Pt.
	25	Rt. Ascen. hours	Fl. Pt.
	26	Observation type	

PPBUF 5 Cells

PPBUF is used in subroutine MARTINI for use in the computation of zonal harmonics. PPBUF contains values of P'_n for n = 1, 2, 3, 4, 5

PPBUF + 0 $P' = 1$

\+ 1 P'_2

\+ 2 P'_3

\+ 3 P'_4

\+ 4 P'_5

PREDBF 11 Cells

PREDBF has two uses:

(1) When the weight and bias information is printed, PREDBF is used to hold information in output format.

PREDBF	+ 0	σ range	(km.)
	+ 1	σ azimuth	(deg.)
	+ 2	σ elevation	(deg.)
	+ 3	σ range rate	(km/sec)
	+ 4	range bias	(km.)
	+ 5	azimuth bias	(deg.)
	+ 6	elevation bias	(deg.)
	+ 7	range rate bias	(km/sec)
	+ 8	time bias	(sec.)

(2) PREDBF is used to hold initial or final conditions as used in the integration buffer. This allows the integration scheme to be restarted with a minimum of computation.

PREDBF	+ 0	time			= W + 1
	+ 1	Δ time			= W + 2
	+ 2	last time			= W + 3
	+ 3	direction			= W + 4
	+ 4	L		-	= W + 5
	+ 5	a_x		x	= W + 6
	+ 6	a_y		y	= W + 7
	+ 7	a_z	or	z	= W + 8
	+ 8	h_x		$\dot{x}$	= W + 9
	+ 9	h_y		$\dot{y}$	= W + 10
	+ 10	h_z		$\dot{z}$	= W + 11

RBUF 9 Cells

RBUF is used in subroutine MARTINI for the computation of tesseral harmonics. RBUF contains entries of $R_{n,m} (U_z)$ for n and m up to 4.

$$R_{n,m} (U_z) = \frac{P_{n,m} (U_z)}{\sqrt{1 - U^2}}$$

RBUF + 0 $R_{1,1} = 1$

+ 1 $R_{2,2}$

+ 2 $R_{3,1}$

+ 3 $R_{3,2}$

+ 4 $R_{3,3}$

+ 5 $R_{4,1}$

+ 6 $R_{4,2}$

+ 7 $R_{4,3}$

+ 8 $R_{4,4}$

RPBUF 8 Cells

RPBUF is used in subroutine MARTINI for the computation of tesseral harmonics. RPBUF contains entries of $R'_{n,m}$ for n and m up to 4.

RPBUF	+ 0	$R'_{2,2}$
	+ 1	$R'_{3,1}$
	+ 2	$R'_{3,2}$
	+ 3	$R'_{3,3}$
	+ 4	$R'_{4,1}$
	+ 5	$R'_{4,2}$
	+ 6	$R'_{4,3}$
	+ 7	$R'_{4,4}$

SBUF   151 Cells

SBUF contains sensor information which was originally stored in SBLOC. SBUF and BIBUF occupy the same locations in core; however, SBUF is not set until BIBUF is no longer needed. After SBUF is set up, the SBLOC area is used for temporaries to save space in core.

SBUF	+ 0	00000SSS, BCD Sensor Number
	+ 1	Ø
	+ 2	λ
	+ 3	X/cos θ
	+ 4	Z
	⋮	Up to 29 more sets of sensor no., Ø, λ, X/cosθ, Z, followed by one BCD word of 00000ZZZ
	+ 150	

SIGN 7 Cells

SIGN contains the standard deviations of the corrections to the elements after each iteration. SIGN is used only for output purposes. If an element is not being corrected, the entry in the buffer is set to octal 0 to suppress printing.

SIGN + 0	$\sigma\, n$
+ 1	$\sigma\, a_{xn}$
+ 2	$\sigma\, a_{yn}$
+ 3	$\sigma\, U_o$
+ 4	σ_{Ω}
+ 5	$\sigma\, i$
+ 6	$\sigma\, B$

SINCOS 8 Cells

SINCOS contains the sines and cosines of λ_E, $2\lambda_E$, $3\lambda_E$, $4\lambda_E$ which are necessary for the computation of tesseral harmonics in the MARTINI subroutine.

SINCOS	+ 0	$\cos \lambda_E$
	+ 1	$\sin \lambda_E$
	+ 2	$\cos 2 \lambda_E$
	+ 3	$\sin 2 \lambda_E$
	+ 4	$\cos 3 \lambda_E$
	+ 5	$\sin 3 \lambda_E$
	+ 6	$\cos 4 \lambda_E$
	+ 7	$\sin 4 \lambda_E$

TAPBUF 128 Cells

TAPBUF is used as an output buffer to accumulate one block of prediction points to be written on the binary ephemeris tape (logical 12).

TAPBUF	+ 0	t (min)
	+ 1	x (e.r.)
	+ 2	y (e.r.)
	+ 3	z (e.r.)
	+ 4	$\dot{x}$ (e.r./ke.)
	+ 5	$\dot{y}$ (e.r./ke.)
	+ 6	$\dot{z}$ (e.r./ke.)
	⋮	Up to 17 sets of t, x, y, z, $\dot{x}$, $\dot{y}$, $\dot{z}$
	+ 126	$\overline{0 \qquad 0}$
	+ 127	$\overline{0 \qquad 0}$

TBUF 15 Cells

TBUF is used in subroutine DIVDIF for temporary storage while interpolating with a fourth order divided difference method. The contents of TBUF are quite variable and cannot be described as single quantities.

TBUF + 0 thru TBUF + 14	Temporaries

TERMS 8 Cells

TERMS contains the coefficients to be included in the least squares solution matrix. These coefficients are computed by CMPCF or CORRD depending on the type of observations. Only the coefficients related to elements being corrected are computed; these are packed into the TERMS buffer in the following order: n, a_{xn}, a_{yn}, U_o, Ω, i, B.

TERMS	+ 0	$C_{\Delta n/n}$
	+ 1	$C_{\Delta a_{xn}}$
	+ 2	$C_{\Delta a_{yn}}$
	+ 3	$C_{\Delta U_o}$
	+ 4	$C_{\Delta \Omega}$
	+ 5	$C_{\Delta i}$
	+ 6	$C_{\Delta B/B}$
	+ 7	R (weighted residual)

TERMS is set up as above when all seven elements are being corrected. If a_{yn} and i were not being corrected TERMS would be:

TERMS	+ 0	$C_{\Delta n/n}$
	+ 1	$C_{\Delta a_{xn}}$
	+ 2	$C_{\Delta U_o}$
	+ 3	$C_{\Delta \Omega}$
	+ 4	$C_{\Delta B/B}$
	+ 5	R (weighted residual)
	+ 6	0
	+ 7	0

W 203 Cells

See Appendix for description.

WBUF 151 Cells

WBUF contains sensor weighting information for range, azimuth, elevation, and range rate. There are about 16 sets of sensor weights assembled in the program; these may be changed or extended by introducing weight and bias cards on logical tape 7.

WBUF + 0 00000SSS, BCD sensor number

+ 1 $1/\sigma_\rho$ (e.r.$^{-1}$)

+ 2 $1/\sigma_A$ (rad.$^{-1}$)

+ 3 $1/\sigma_h$ (rad.$^{-1}$)

+ 4 $1/\sigma_{\dot{\rho}}$ (ke/e.r.)

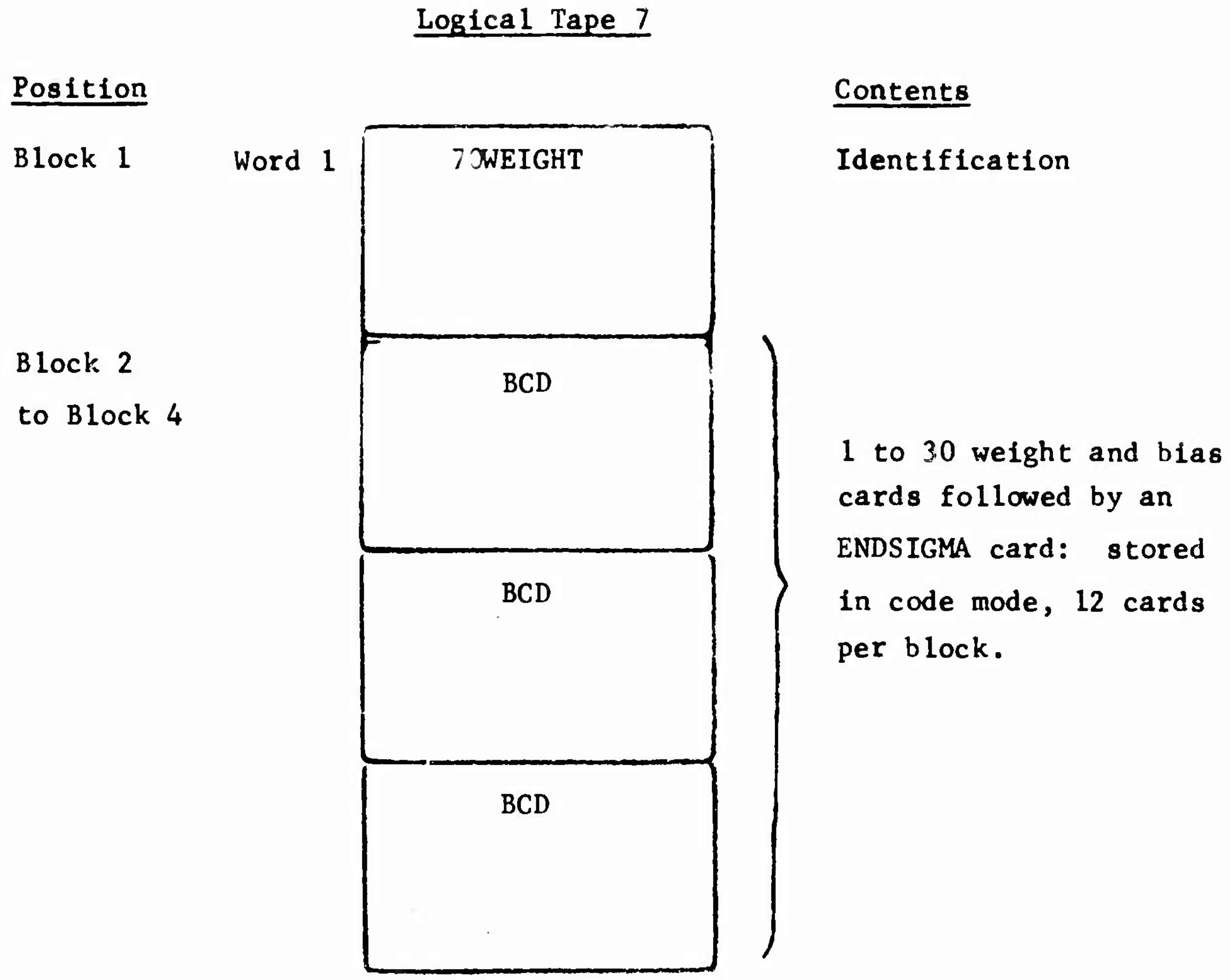
$\left.\begin{array}{c}\cdot\\\cdot\\\cdot\\\cdot\\\cdot\\\cdot\end{array}\right\}$ Up to 29 sets of sensor number, $1/\sigma_\rho$, $1/\sigma_A$, $1/\sigma_h$, $1/\sigma_{\dot{\rho}}$, followed by 1 BCD word of 00000ZZZ.

+ 150

Z2BUF 7 Cells

Z2BUF is used in the CDLTBIN subroutine for temporary storage while computing Bessel Functions for $\Delta B/B$ correction. Z2BUF is used for two different purposes, depending on the magnitude of $Z/2$.

	For $Z/2 < 1$	For $Z/2 \geq 1$
Z2BUF + 0	1	1
+ 1	$(Z/2)^2/1!$	$2/Z$
+ 2	$(Z/2)^4/2!$	$(2/Z)^2$
+ 3	$(Z/2)^6/3!$	$(2/Z)^3$
+ 4	$(Z/2)^8/4!$	$(2/Z)^4$
+ 5	$(Z/2)^{10}/5!$	$(2/Z)^5$
+ 6	blank	$(2/Z)^6$



1 to 30 weight and bias cards followed by an ENDSIGMA card: stored in code mode, 12 cards per block.

FIGURE 3. SENSOR WEIGHT AND BIAS TAPE FORMAT

Logical Tape 12

Position			Contents	
Block 1	Word 1	BCD		
	Word 2	BCD		
Block 2 to Block N-1	1	48 bit floating point	t	Time since epoch (minutes)
	2	48 bit floating point	x	(earth radii)
	3	48 bit floating point	y	(earth radii)
	4	48 bit floating point	z	(earth radii)
	5	48 bit floating point	$\dot{x}$	(e.r./ke)
	6	48 bit floating point	$\dot{y}$	(e.r./ke)
	7	48 bit floating point	$\dot{z}$	(e.r./ke)
			17 more sets of t, $\underline{r}$, $\underline{\dot{r}}$ followed by 2 words of fixed point 0	
			Up to 18 sets of t, $\underline{r}$, $\underline{\dot{r}}$ followed by 1 word of alpha-numeric Z's	

FIGURE 4. BINARY EPHEMERIS TAPE FORMAT

3.3 MEMORY STORAGE DIAGRAM

	ASSIGNMENT	NO. OF WORDS	OCTAL LOCATION
EXECUTIVE	EXECMOD1 EXECMOD2 MASTER ASSIGN AREA	6707_{10}	00000
SYSTEM FILES	EBLOC	6400_{10}	15064
	OBLOC	5000_{10}	31464
	SBLOC	1388_{10}	43274
PROGRAM	SPIRDECA	6708_{10}^{*}	46050
BMEWS AREA	BMEWS	6500_{10}	63234

* 6708 does not include 785 locations assigned within SBLOC that are used by SPIRDECA after initialization.

FIGURE 5. MEMORY STORAGE DIAGRAM

SECTION 4

SUBROUTINE DESCRIPTIONS

This section contains a description of each subroutine used by the Spiral Decay Program. Each subroutine description lists the purpose, usage, subroutines used, size, description; flow charts are included when necessary.

4.1 ALPHABETICAL INDEX TO SUBROUTINE DESCRIPTIONS

Name	Function	Page
ADBASH	Numerical Integration Routine	70
ALREC	Compute $\underline{A}$, $\underline{D}$, $\underline{L}$ from α, δ	71
ANGSUN	Compute α, δ of the sun	72
AZREC	Compute $\underline{A}$, $\underline{D}$, $\underline{L}$ from a, h	74
BCDTIM	Convert Time to BCD Date	76
BEGIN	Compute classical elements at Epoch	77
BIAS	Apply Biases to the Observations	80
CALH	Compute Height	81
CALU	Compute $\underline{U}$	82
CARDER	Output Card Error Comments	83
CDERIV	Derivative Routine for Cowell Integration	84
CDLTB	Compute $C\Delta B/_{B}$	86
CDLTBIN	Initialize Subroutine CDLTB	89

Name	Function	Page
CHECK	Test for Convergence	91
CHECKI	Initialize Subroutine CHECK	93
CHGCWD	Modify Output Routine PRNTMAT	94
CHGNXN	Setup Least Squares Matrix	95
CMPCF	Compute Coefficients for ρ, A, h, $\dot{\rho}$	96
CNTRL	Differential Correction Control	98
COEFF2	Control Coefficient Computation	105
COELTS	Convert Elements for Output	107
COMDEL	Compute $\Delta \underline{L}$	108
COMDQ	Compute Δq	109
COMRMS	Compute Root-mean-square	110
COMXLX	Compute $\underline{L}$	111
CORRD	Compute Range Rate Coefficients	112
CREKT	Apply Corrections to the Elements	114
DELOUT1	Print Delta Elements	116
DELOUT2	Print Final Results	117
DELTAU	Compute Δu	118
DERIV	Compute Derivatives	119
DIVDIF	Divided Difference Interpolation	120
DOTPR	Compute Dot Products	122
ELMOUT	Print Corrected Elements after Convergence	123
ELMOUT1	Print Corrected Element after Each Pass	124
FTAPEW	Wrap-Up Ephemeris Tape	126
GETSEN	Retrieve Sensor Information	127
GETWGT	Retrieve Weighting Information	128
GIPAR	Store Information for Program GIPAR	129
HEAD	Output Page Headings	130

Name	Function	Page
IHEAD1	Initialize for Page Headings	131
INITIAL	Initialize for Integration	132
INITL	Print and Process Cards	133
INPUT	Unpack and Validate P Cards	134
ITAPEW	Initialize Ephemeris Tape	144
JNDRAG	Compute Drag Perturbations	145
LPART	Compute L Modulo 2π	149
LSQ	Clear Least Squares Matrix	150
LSQR	Build Least Squares Matrix	151
LSQS	Solve Least Squares Matrix	152
MARTINI	Compute Bulge Perturbations	153
MATRIX	Compute Correlation Matrix	158
MOD2PI	Modulo 2π Subroutine	160
MOVBUF	Move Observation Buffer	161
MOVDAT	Retrieve Processed Observations	163
NXTOB	Retrieve Observations in OBLOC Format	165
OBVEC	Reformat Observations	167
PAGECON	Page Control for Output	169
PCONTRL	Prediction Control	170
PHLAH	Compute ϕ, λ, h	182
PRERES	Compute New Epoch Elements	183
PRINTW	Print Weights and Biases	186
PROOBS	Reformat Observation Buffer	187
PRNTMAT	Print Correlation Matrix	189
RDPRES	Compute Radiation Pressure	190
RDTSB	Compute Range Rate	191
READOBS	Read Observations from Tape	192

Name	Function	Page
REJECT1	First Pass Rejection	193
REJECT2	Range Rate Rejection	194
RESICK	Restore Integration Buffer	195
RESOUT1	Output Residuals	196
RESOUT2	Convert Time to BCD for Residuals	198
RESOUT4	Print Page Headings for Residuals	198
RESREJ1	Residual Rejection	199
RESW	Restore Integration Buffer	200
RESWBF	Restore Integration Buffer	201
REVSUB	Update Revolution Number	202
RHOSB	Compute ρ	203
RINEL	Restore Initial Elements	204
RR2AHL	Convert Elements	205
SAVCON	Save Card Images	207
SAVELM	Save Elements	208
SAVICK	Save ICK Buffer	209
SAVW	Save Integration Buffer	210
SAVWBF	Save Integration Buffer	211
SAV5PTS	Save Elements for Interpolation	212
SENLOC	Compute Sensor Location	213
SETSBUF	Store Sensor Information	214
SETW	Set Buffer for SUBXYZ	216
SINEL	Save Initial Elements	217
SORTOB	Sort Observations by Time	218
SUBOUT	Output Prediction Ephemeris	219
SUBOUTI	Initialize for Output	220
SUBXYZ	Compute Position and Velocity	221

Name	Function	Page
TAPEW	Write Ephemeris Tape	224
TEMP	Compute Epoch Temperature	225
THGRC	Compute θ_{GR}	226
WEIGHT	Read Weights	227
WILBUR	Print Observations	229
WSETUP	Set Integration Buffer	230

4.2 SUBROUTINE DESCRIPTIONS

ADBASH
SPIRDEC

PURPOSE:	To integrate a system of seven first order differential equations from $X = X_n$ to $X = X_{n+1}$: $Y_i' = f_i (X, Y_1, Y_2, \ldots, Y_7); i = 1, 2, \ldots, 7$
CALL SEQUENCE:	After initialization (see "How to use ADBASH"), to start or continue integration simply, α JMP ADBASH α + 1H Error Return α + 2H Normal Return
INPUT:	$Y_1, Y_2, Y_3, \ldots, Y_7$ and X, for $X = X_n$
OUTPUT:	$Y_1, Y_2, Y_3, \ldots, Y_7$ and X, for $X = X_{n+1}$
SUBROUTINES:	Program: LPART, DERIV, CDERIV
STORAGE REQUIREMENTS:	520 decimal or 1010 octal locations.
METHOD:	The Adams-Bashforth (A-B) method, together with the Runge-Kutta (R-K) method of integration are employed in this subroutine. R-K is used as a starting procedure so that an adequate number of step-wise solutions may be obtained to build a difference table needed by the A-B method. The interval of integration is automatically varied to keep the discrepancy between the integrated values and an absolute error check within prescribed limits.
REFERENCE:	A detailed description of this subroutine is included in the Appendix section of this manual.

PURPOSE:	To compute $\underline{A}$, $\underline{D}$, $\underline{L}$ if α, δ are observed.
CALL SEQUENCE:	JMP ALREC
INPUT:	ALPHA = α (rad) DELTA = δ (rad)
OUTPUT:	ASUBX = -sin (ALPHA) ASUBY = cos (ALPHA) ASUBZ = F/0 DSUBX = [-sin (DELTA)] x [ASUBY] DSUBY = [sin (DELTA)] x [ASUBX] DSUBZ = cos (DELTA) XLSUBX = [cos (DELTA)] x [ASUBY] XLSUBY = [-cos (DELTA)] x [ASUBX] XLSUBZ = sin (DELTA)
SUBROUTINES:	PHILCO - FSIN, FCOS
STORAGE REQUIREMENTS:	15 Cells
DESCRIPTION:	Computes the values listed under Output. Called by subroutine MOVDAT.

PURPOSE: To compute the sun's position at time t.

CALL SEQUENCE: JMP ANGSUN

INPUT:

W + 1 = t (minutes since epoch)

EPOCH = days and frac from 1950 to epoch

XLSUNO = L_{0_c} at beginning of year

C1 = .985647346 deg/day

C2 = 1.91633 deg

C3 = C_3 - from TLC subroutine

C4 = 2.578909

C5 = .4336428

OUTPUT:

XLSUNT = L_{Θ} at time t

SUNLX = L_x

SUNLY = L_y

SUNLZ = L_z

ALSUN = α_{Θ}

DLSUN = δ_{Θ}

CSALS = cos α_{Θ}

SNALS = sin α_{Θ}

CSDLS = cos δ_{Θ}

SNDLS = sin δ_{Θ}

SUBROUTINES: PHILCO - FSIN, FCOS, FATAN

STORAGE REQUIREMENTS: 23 Cells

DESCRIPTION: Computes the following:

$$L_{\Theta_t} = C_1 (t) + L_{\Theta_o} + C_2 \sin \left(\frac{C_1 \quad t+C_3}{57.295...} \right)$$

where t = days since Jan 0.0

DESCRIPTION:
(continued)

$$\alpha_{\Theta_t} = - \frac{1}{57.295\ldots} \left\{ C_4 \sin \left(\frac{2L_{\Theta_t}}{57.295} \right) - L_{\Theta_t} \right\}$$

$$\alpha_{\Theta_t} = \tan^{-1} \left\{ C_5 \sin \alpha_{\Theta} \right\}$$

$$\mathrm{SUNLX} = \cos \alpha_{\Theta} \cos \delta_{\Theta}$$

$$\mathrm{SUNLY} = \cos \delta_{\Theta} \sin \alpha_{\Theta}$$

$$\mathrm{SUNLZ} = \sin \delta_{\Theta}$$

AZREC
SPIRDEC
1 of 2

PURPOSE: To compute A, D, L if azimuth and elevation are observed.

CALL SEQUENCE: JMP AZREC

INPUT:

ALPHA = A (rad)
DELTA = h (rad) PHIRD = 0
COSTH = cos θ
SINTH = sin θ

OUTPUT:

ASUBX
ASUBY
ASUBZ
DSUBX
DSUBY
DSUBZ
XLSUBX
XLSUBY
XLSUBZ

SUBROUTINES: PHILOC - FSIN, FCOS

STORAGE REQUIREMENTS: 61 Cells

DESCRIPTION: Computes A, D, L from the following formulation. Used by subroutine MOVDAT.

$$\left.\begin{array}{l} \mathrm{ASUBXT} = \sin\ (\mathrm{ALPHA}) \\ \mathrm{ASUBYT} = \cos\ (\mathrm{ALPHA}) \\ \mathrm{ASUBZT} = \mathrm{F/O} \end{array}\right\} A_t$$

$$\left.\begin{array}{l} \mathrm{XLSUBXH} = -\cos\ (\mathrm{DELTA})\ \cos\ (\mathrm{ALPHA}) \\ \mathrm{XLSUBYH} = \cos\ (\mathrm{DELTA})\ \sin\ (\mathrm{ALPHA}) \\ \mathrm{XLSUBZH} = \sin\ (\mathrm{DELTA}) \end{array}\right\} L_h$$

$$\left.\begin{array}{l} \mathrm{DSUBXT} = \sin\ (\mathrm{DELTA})\ \cos\ (\mathrm{ALPHA}) \\ \mathrm{DSUBYT} = -\sin\ (\mathrm{DELTA})\ \sin\ (\mathrm{ALPHA}) \\ \mathrm{DSUBZT} = \cos\ (\mathrm{DELTA}) \end{array}\right\} D_t$$

SINPH = sin 0
COSPH = cos 0

DESCRIPTION:
(continued)

$$\left.\begin{array}{l}\text{ZSUBX} = \cos \phi \cos \theta \\ \text{ZSUBY} = \cos \phi \sin \theta \\ \text{ZSUBZ} = \sin \phi\end{array}\right\} Z$$

$$\left.\begin{array}{l}\text{SSUBX} = \sin \phi \cos \theta \\ \text{SSUBY} = \sin \phi \sin \theta \\ \text{SSUBZ} = -\cos \phi\end{array}\right\} S$$

$$\left.\begin{array}{l}\text{ESUBX} = -\sin \theta \\ \text{ESUBY} = \cos \theta \\ \text{ESUBZ} = \text{F/O}\end{array}\right\} E$$

$$\text{XLSUBX} = L_{xh}S_x + L_{yh}E_x + L_{zh}Z_x$$
$$\text{XLSUBY} = L_{xh}S_y + L_{yh}E_y + L_{zh}Z_y$$
$$\text{XLSUBZ} = L_{xh}S_z + L_{yh}E_z + L_{zh}Z_z$$

$$\text{ASUBX} = A_{xt}S_x + A_{yt}E_x + A_{zt}Z_x$$
$$\text{ASUBY} = A_{xt}S_y + A_{yt}E_y + A_{zt}Z_y$$
$$\text{ASUBZ} = A_{xt}S_z + A_{yt}E_z + A_{zt}Z_z$$

$$\text{DSUBX} = D_{xt}S_x + D_{yt}E_x + D_{zt}Z_x$$
$$\text{DSUBY} = D_{xt}S_y + D_{yt}E_y + D_{zt}Z_y$$
$$\text{DSUBZ} = D_{xt}S_z + D_{yt}E_z + D_{zt}Z_z$$

PURPOSE:	To convert time (min since epoch) to BCD date.
CALL SEQUENCE:	TMA (Time in min.) JMP BCDTIM
INPUT:	EPOCH = days and fractions since 1950
OUTPUT:	OBYEAR = YY OBMO = MM OBDAY = DD } BCD left justified FOBHR = F/hours FOBMIN = F/min FOBSEC = F/sec
SUBROUTINES:	System - DKLOK, SEPSUB
STORAGE REQUIREMENTS:	17 Cells
DESCRIPTION:	Time is converted to days and added to EPOCH. This time is converted to BCD (YYMMDD), by DKLOK subroutine. The fraction of a day remaining is then converted to floating pt. hours, min., sec.. If the seconds $\geq$ 59.9 the time is rounded to the nearest minute.

PURPOSE: To compute initial orbit conditions from the input elements.

CALL SEQUENCE: JMP BEGIN

INPUT:

HXO = h_{xo}
HYO = h_{yo}
HZO = h_{zo}
AXNO = a_{xno}
AYNO = a_{yno}
XLO = L_o
XKERTM = k_e

OUTPUT:

P = p
RTP = $\sqrt{p}$
WX = W_x
WY = W_y
WZ = W_z
COSI = $\cos i$
SINI = $\sin i$
XINCL = i
SINO = $\sin \Omega$
XNODEO = Ω
ESQ = e_o^2
EO = e_o
AO = a_o
RTA = $\sqrt{a_o}$

UO = U_o
XNO = n_o
AXO = a_{xo}
AYO = a_{yo}
AZO = a_{zo}
QO = q_o

SUBROUTINES:

Program - ARCTAN
Philco - FSQRT

STORAGE REQUIREMENTS: 43 Cells

DESCRIPTION: $p = \sqrt{h_{xo}^2 + h_{yo}^2 + h_{zo}^2} \longrightarrow P$

$\sqrt{p} \longrightarrow RTP$

$W_x = \frac{h_{xo}}{\sqrt{p}} \longrightarrow WX, \quad x \rightarrow y,z$

$\cos i = W_z \longrightarrow COSI$

$\sin i = \sqrt{1-\cos^2 i} \longrightarrow SINI$

$i = \tan^{-1} \left[\frac{\sin i}{\cos i}\right] \longrightarrow XINCL$

$\sin \Omega = W_x / \sin i \longrightarrow SINO$

$\cos \Omega = W_y / \sin i \longrightarrow COSO$

$\Omega = \tan^{-1} \left[\frac{\sin \Omega}{\cos \Omega}\right] \longrightarrow XNODEO$

$e_o^2 = a_{xno}^2 + a_{yno}^2 \longrightarrow ESQ$

$e_o = \sqrt{e^2} \longrightarrow EO$

$a_o = p/(1-e^2 \longrightarrow AO$

$\sqrt{a_o} \longrightarrow RTA$

$U_o = L_o - \Omega \longrightarrow UO$

$n_o = k_e \sqrt{E} / a_o^{3/2} \longrightarrow XNO$

DESCRIPTION:
(continued)

$$a_{xo} = -\sin\Omega \cdot \cos i \cdot a_{yno} + \cos\Omega \, a_{xno} \longrightarrow \text{AXO}$$

$$a_{yo} = +\cos i \cdot a_{yno} \cdot \cos\Omega + \sin\Omega \, a_{xno} \longrightarrow \text{AYO}$$

$$a_{zo} = \sin i \cdot a_{yno} \longrightarrow \text{AZO}$$

$$q_o = a_o \; (1 - e_o) \longrightarrow \text{QO}$$

PURPOSE: To subtract biases from the observations.

CALL SEQUENCE: JMP BIAS
(Return)

INPUT: BIASAD = C/HLT, EBLOC; C/HLT, BIBUF

BIBUF = buffer of biases stored by sensor

Observations starting in location EBLOC

OUTPUT: See description

SUBROUTINES: None

STORAGE
REQUIREMENTS: 22 Cells

DESCRIPTION: The sensor number is extracted from an entry in the observation buffer and BIBUF is searched for a match.

If a match is found, the range, angle, and range-rate biases are subtracted from the observed quantities. Convert the time of the observation to minutes since epoch, subtract the time bias, and store in same cell (2,1). If bit 0 of the word containing the station number is 1, then the VERLORT/PRELORT corrections are applied. Keep a count of negative observation times in the right address of MCOUNT for the SORTOB subroutine. If no match is found, proceed to the next observation without any error indication.

This procedure is continued until the sentinel of Z's is found at the end of the observation buffer.

PURPOSE: To compute height above sea level.

CALL SEQUENCE: JMP CALH

INPUT:
$UX = U_z$
$R = r$
$F = f = 1/298.3$

OUTPUT: $H = h$ (e.r.)

SUBROUTINES: (None)

STORAGE REQUIREMENTS: 8 Cells

DESCRIPTION: $H = h = r-1 + U_z^2 F + 3/2 F^2 U_z^2 (1-U_z^2)$

PURPOSE: To compute $\underline{U}$.

CALL SEQUENCE:

```
TMD C/HLT, W + 5; C/HLT, X
JMP  CALU
```

INPUT:

$W + 5 = x$

$+ 6 = y$

$+ 7 = z$

$+ 8 = \dot{x}$

$+ 9 = \dot{y}$

$+10 = \dot{z}$

OUTPUT:

R = r

UX = U_x

UY = U_y

UZ = U_z

X = x

Y = y

Z = z

XDOT = $\dot{x}$

YDOT = $\dot{y}$

ZDOT = $\dot{z}$

SUBROUTINES: PHILCO = FSQRT

STORAGE REQUIREMENTS: 12 Cells

DESCRIPTION: Moves values from W + 5 thru W + 10 to X, Y, Z, XDOT, YDOT, ZDOT.

Computes the following:

$$r = \sqrt{x^2 + y^2 + z^2}$$

$$U_x = x/r$$

$$U_y = y/r$$

$$U_z = z/r$$

CARDER
SPIRDEC

PURPOSE: To output card error comments.

CALL SEQUENCE: (4 entrances)

```
(1)  TMA  (Word)            (3)  JMP CARDERB
     JMP  CARDER

(2)  TMA  (Word)            (4)  TMA  (Word)
     JMP  CARDERA                JMP  CARDERC
```

INPUT: Word = BCD at T47
XR3 = location of 1st word of the erroneous card

OUTPUT: Comment on hard copy - (see description)

SUBROUTINES: System - PANT
Program - PAGECON

STORAGE REQUIREMENTS: 44 Cells

DESCRIPTION: The subroutine will insert the given parameter into one of the following, corresponding to the entrance:

(1) "Error in field ending in Col. XX"

(2) "System expects card with X in Col. 80"

(3) (No Comment)

(4) "Card repeated, too many cards of type X."

The program will then output the comment (or no comments) followed by:

1 8 16 24 32 40 48 56 64 72 80

(80 Column Card Image ————————→)

1 Line Space

Cell CARDSW is set ≠ 0 to indicate this subroutine has been used.

PURPOSE:	To compute the derivatives of position and velocity elements with respect to time, using the effects of radiation pressure, drag and/or bulge perturbations as specified on P Cards.
CALL SEQUENCE:	JMP CDERIV
INPUT:	W + 1 = time since epoch (min.) 5 = F/0 6 = x 7 = y 8 = z 9 = $\dot{x}$ 10 = $\dot{y}$ 11 = $\dot{z}$ And input needed for subroutines listed.
OUTPUT:	W + 187 = F/0 188 = dx/dt XDD = $\ddot{x}$ 189 = dy/dt YDD = $\ddot{y}$ 190 = dz/dt ZDD = $\ddot{z}$ 191 = dx/dt XDGR = $\dot{x}'$ 192 = dy/dt YDGR = $\dot{y}'$ 193 = dz/dt ZDGR = $\dot{z}'$
SUBROUTINES:	Program - CALC, MARTINI, JNDRAG, RDPRES
STORAGE REQUIREMENTS:	25 Cells
DESCRIPTION:	This subroutine is called only when executing the Cowell integration. The perturbative effects are computed by the subroutines, MARTINI, JNDRAG, RDPRES. The derivatives of the elements are then computed and stored in the Adams-Bashforth buffer (W + 187 to W + 193).

FORMULATION: XDD = $\ddot{x} = \mu x/r^3$ $\quad x \longrightarrow y,z$

XDGR = $\dot{x}^{\backprime} = \dot{x}^{\backprime}_B + \dot{x}^{\backprime}_D - \ddot{x}$ $\quad x \longrightarrow y,z$

W + 187 = F/0

188 = $dx/dt = k_e \, \dot{x}$

189 = $dy/dt = k_e \, \dot{y}$

190 = $dz/dt = k_e \, \dot{z}$

191 = $dx/dt = k_e \, \dot{x}^{\backprime}$

192 = $dy/dt = k_e \, \dot{y}^{\backprime}$

193 = $dz/dt = k_e \, \dot{z}^{\backprime}$

PURPOSE: To compute scalar differential expression for $B = C_{[illegible]}\ A$ [illegible]

CALL SEQUENCE:

For range and angles:	For range rate:
JMP CDLTB	JMP CDLTBA
TAM $(C_{\Delta B/B})$	TAM $(C_{\Delta B/B})$

INPUT: Output of subroutines: COEFF2, CORRD, CDLTBIN

OUTPUT: (A) reg = $C_{\Delta B/B}$ and output listed under description.

SUBROUTINES: Program - CORRD

Philco - FEX

STORAGE REQUIREMENTS: 106 cells

DESCRIPTION: This subroutine must be initialized by CDLTBIN subroutine. There are two entrances to the program: (1) for range and angles - JMP CDLTB, (2) for range rate - JMP CDLTBA.

Formulation:

For range and angles, if any of n/n or a_{xn} or a_{yn} is not being corrected, its coefficient will be computed as follows:

$$\text{CDLTN} = C_{\Delta n/n} = \left[U_N \,(L_i \cdot V) + R_N \,(L_i \cdot U) \right] \frac{1}{\sigma}$$

$$\text{CDLTAXN} = C_{\Delta axn} = \left[U_{xn} \,(L_i \,.\, V) + R_{xn} \,(L_i \,.\, U) \right] \frac{1}{\sigma_i}$$

$$\text{CDLTAYN} = C_{\Delta ayn} = \left[U_{xn} \,(L_i \,.\, V) + R_{yn} \,(L_i \,.\, U) \right] \frac{1}{\sigma_i}$$

where i = [illegible], A or h

and $\underline{L}_i$ is $\underline{L}$, $\underline{A}$ or $\underline{D}$ corresponding to the observed quantity.

DESCRIPTION: (continued)

If the range rate entrance is taken and the 3 elements are not being corrected, the coefficients will be computed by using part of the CORRD subroutine. The values will be stored in CDLTN, CDLTAXN, CDLTAYN.

However, if the elements are being corrected for ρ, $\dot{\rho}$, A, or h, the values will be unpacked from TERMS, TERMS+1, and TERMS+2 respectively.

Then compute the following.

$$\text{COS2OM} = \cos 2\omega \left\{ \begin{array}{ll} = 2\left(\dfrac{a_{xno}}{e_o}\right)^2 - 1 & \text{if } e_o \neq o \\ = 0 & \text{if } e_o = 0 \end{array} \right.$$

$$\text{SIN2OM} = \sin 2\omega = \sqrt{1 - \cos^2 2\omega}$$

Set up buffer ALFBUF as powers of α with α defined as:

$$\alpha = -(\sin^2 i)\ (f/2h)\ \cos 2\omega$$

where f flattening = $^1/298.3$

h = perigee height

$$\text{ALFBUF} = \alpha^0/0! = F/1$$

$$\text{ALFBUF+N} = \frac{\alpha^n}{n!} \qquad 1 \leq n \leq 6$$

Compute the following, which will be used to solve for $\delta\omega/_{\delta t}$ and $\delta e/_{\delta t}$:

$$\text{PARTA} = B_o + \alpha B_1 + \frac{\alpha^2}{2!}\ (3\cdot B2) + \frac{\alpha^3}{3!}\ (3\cdot 5B_3)\ldots+\frac{\alpha^6}{6!}\ (3\cdot 5\cdot 7\cdot 9\cdot 11B_6)$$

$$\text{PARTA2} = (B_1 - 1/2\cdot 3\ B_2) + 2(3B_2 - 1/2\cdot 3\ 5B_3)\ldots.+\frac{\alpha^4}{4!}(3\cdot 5\cdot 7\cdot 9B_5 - 1/2\cdot 3\cdot 5\cdot 7\cdot 9\cdot 11B_6)$$

DESCRIPTION:
(continued)

$$\text{PARTA3} = (3B_2 - 3\cdot 5B_3 - 1/4\cdot 3\cdot 5\cdot 7B_4) + \ldots + \frac{\alpha^2}{2!}(3\cdot 5\cdot 7B_4 - 3\cdot 5\cdot 7\cdot 9B_5 + 1/4\cdot 3\cdot 5\cdot 7\cdot 9\cdot 11B_6)$$

$$\text{PARTE3} = B, + \alpha B_2 + \frac{\alpha^2}{2!} 3\cdot B_3 + \ldots + \frac{\alpha^5}{5!} 3\cdot 5\cdot 7\cdot 9B_6$$

$$\text{PARTE2} = (B_2 - 1/2\ 3\cdot B_3) + \alpha(3\cdot B_3 - 1/2\ 3\cdot 5B_4) + \ldots. + \frac{\alpha^3}{3!}(3\cdot 5\cdot 7B_5 - 1/2\ 3\cdot 5\cdot 7\cdot 9B_6)$$

$$\text{PARTE} = (3B_3 - 3\cdot 5B_4 - 1/4\cdot 3\cdot 5\cdot 7B_5) + \alpha(3\cdot 5B_4 - 3\cdot 5\cdot 7B_5 + 1/4\cdot 3\cdot 5\cdot 7\cdot 9B_6)$$

where 1) powers of α are in ALFBUF

2) $1\cdot 3 \ldots\ldots (2N-1)\ B_n$ are in BNNBUF

3) $1\cdot 3 \ldots\ldots (2N-3)\ B_n$ are in BNBUF

Solve for $\delta a/\delta t$ and $\delta e/\delta t$:

$$\frac{\delta a}{\delta t} = \exp(-z)\left[(\text{PARTA}) + (\alpha\tan 2\omega)^2(\text{PARTA2}) + \frac{(\alpha\tan 2\omega)^4}{6}(\text{PARTA3})\right]$$

$$\frac{\delta e}{\delta t} = \exp(-z)\left[(\text{PARTE}) + (\alpha\tan 2\omega)^2(\text{PARTE2}) + \frac{(\alpha\tan 2\omega)^4}{6}(\text{PARTE3})\right]$$

Solve for $C_{\Delta B/B} = (M-M_o)\frac{(aBo\pi)}{2}\left[3C_{\Delta n/n}(\delta a/\delta t) - (C_{\Delta axn}\,a_{xn} + C_{\Delta ayn}\,a_{yn})\left(\frac{p}{h}\right)\left(\frac{\delta e}{\delta t}\right)\right]$

PURPOSE: To initialize for CDLIB subroutine.

CALL SEQUENCE: JMP CDLIBIN

INPUT: SKNTRL = 0 ‾‾‾‾ 0XXXXXXX

ASCON2 = ?/6378165 (m/er)

Input needed for SUBXYZ subroutine

OUTPUT: See description

SUBROUTINES: Program - SUBXYZ, JNDRAG, MULTSUB

STORAGE REQUIREMENTS: 96 Cells

DESCRIPTION: Test if correcting B; if not exit. Use part of JNDRAG subroutine to compute S and log ρ (h_i, T), i = 1, 2, 3, 4 and perigee density (ρ_π).

Then compute density scale height:

$$HSUBS = \frac{1}{h_s} = 6378.165 \ \frac{\log_e 10}{10} \left[\frac{(3S^2-6S+2)}{6} \log \rho \ (h_1, T) - \frac{(3S^2-4S-1)}{2} \log \rho \ (h_2, T) + \frac{(3S^2-2S-2)}{2} \log \rho \ (h_3, T) - \frac{(3S^2-1)}{6} \log \rho \ (h_4, T) \right]$$

Compute constant values needed for CDLIB:

ABRHO = $(a B\rho_\pi) / 2$

POVH = p/h_s

AOVH = a/h_s

ZZ = $z = ae/h_s$

DESCRIPTION: (continued)

Test z/2:

If $z/2 < 1$, set Z2BUF as follows:

Z2BUF = F/1

$$\text{Z2BUF} + r = \frac{(z/2)^{2r}}{r!} \qquad r = 1, 2, \ldots, 5$$

Then solve for B_n,

$$B_n = \sum_{r=0}^{5} \frac{(z/2)^{2r}}{(r!)\ (n+r)!} \qquad n = 1, 2, \ldots 6$$

If $z/2 > 1$:

Z2BUF = 1

$$\text{Z2BUF}+x = \left(\frac{2}{z}\right) \frac{1}{\sqrt{2\pi z}} \quad , \quad x = 1, \ldots, 5$$

Set up BNBUF and BNNBUF:

$$e^{-z} B_n = \left(\frac{2}{z}\right)^n \frac{1}{\sqrt{2\pi z}} \Sigma\ L_{n,m} \qquad \text{where } n = 1, \ldots, 6$$

$$\text{where } L_{n,o} = 1 \text{ and } L_{n,m} = L_{n,m-1} \left[\frac{(m-{}^1/2-n\quad m-{}^1/2+n)}{2mz}\right]$$

The series should be truncated when:

$$\frac{L_{n,\ m}}{L_{n,\ m-1}} > 1$$

Whatever the value of Z/2, set switch JMPTBL + 3 = C/JMP, PA1; C/JMP, SOLPRTA in CDLTB subroutine.

PURPOSE: To check for correction cycle convergence.

CALL SEQUENCE: JMP CHECK

INPUT:

RPT = 0———0X, X = No. of iterations

RCNT = No. of residuals

RMS = Current root mean square of residuals

OLDRMS = Previous RMS

DQFLAG = Flag for Δq check

DQN = Δq max

CONTEST = ϵ for allowable RMS change for convergence

OUTPUT: See description.

SUBROUTINES: Program - CHECK1, COMDQ

STORAGE REQUIREMENTS: 65 Cells

DESCRIPTION: This is not a closed subroutine in all cases. If the differential correction should continue, the subroutine exit employs the Jump Register (closed). If the D.C. converges, the exit is to location CONCOM; if the D.C. diverges, to location DIVCOM.

(1) At CHSW+1H, the second pass RMS must be saved for output at the end of the D.C. if the first pass is an "n only" correction. Otherwise, the first pass RMS is saved.

(2) At CHQ:

If a_{xn} and a_{yn} are not being corrected, there is no need for a Δq check. If they are being corrected, and a Δq test is requested, the following procedure is followed

DESCRIPTION:
(continued)

(a) Δq is computed by COMDQ subroutine

(b) q may increase by any amount but it may only decrease by Δq max (from P Card 3). If the decrease $\geq \Delta q$ max, Δa_{xn} and Δa_{yn} are modified by 1/2 and the test is repeated until $\Delta q < \Delta q$ max.

(3) At CHRMS, test the RMS:

(a) Compute the change in the RMS. If it is less than CONTEST (usually .01),then convergence has occurred.

(b) If the change is greater, exit to CONCOM, test if the current RMS is greater than the previous one. If it is, the correction is diverging. If the RMS increases on 4 consecutive iterations, exit to DIVCOM.

(c) At CHSW2:
If the change is greater than CONTEST, but the RMS is smaller than the previous one, the number of iterations (location RPT) is decreased by 1 and tested for 0. If RPT $\neq$ 0, the D.C. will continue.

(d) At CCC200:
If RPT = 0 for the first time, a comment will be printed stating that CONTEST will be multiplied by 5 and the D.C. will be repeated for the same number of iterations.

(e) At CCC190:
If RPT = 0 for the second time, the run will be terminated by a comment stating that convergence is not determined.

CHECKI
SPIRDEC

PURPOSE: To initialize subroutine CHECK

CALL SEQUENCE: JMP CHECKI

INPUT: SKNTRL = $0\overline{\qquad\quad}0$XXXXXXX

the last 7 bits correspond respectively to B, i, Ω, U_o, a_{xn}, a_{yn}, n

If the bit is = 1, the element is to be corrected.

CNFLAG = 1, then do n only correction first.

OUTPUT: See description.

SUBROUTINES: Program - CHGNXN

STORAGE REQUIREMENTS: 14 Cells

DESCRIPTION: (1) Sets the following switches in the CHECK subroutine:

CHSW2 = JMP CCC200
CHSW1 = JMP CHQ
CHSW = JMP CHSW+1H

(2) Moves SKNTRL to KNTRL. Tests if n only correction, if yes:
Set KNTRL = $0\overline{\qquad\quad}0$1 and CHSW = JMP CHN

(3) Sets OLDRMS = F/1000000
DIVFL = $0\overline{\qquad\quad}0$
COUNTL = C/HLT, N,where N = number of bits = 1 in KNTRL

(4) Uses subroutine CHGNXN to set up matrix size.

PURPOSE:	To modify the output control words for subroutine PRNTMAT.
CALL SEQUENCE:	JMP CHGCWD
INPUT:	COUNTR = C/HLT,O; C/HLT,N N = matrix size $1 \leq N \leq 7$
OUTPUT:	Modified control words for subroutine PRNTMAT.
SUBROUTINES:	None
STORAGE REQUIREMENTS:	19 cells
DESCRIPTION:	If the matrix to be printed by subroutine PRNTMAT is a 1 x 1 or a 7 x 7, no modification of the output control words is necessary. If the matrix is a 2 x 2 to a 6 x 6, the control words must be modified to pick up the correct locations in buffer MATRIXB. (See subroutine MATRIXB for definition of buffer.)

PURPOSE: To set up parameters for least squares subroutines.

CALL SEQUENCE:
TMA C/HLT,N N = matrix size
JMP CHGNXN

INPUT: (A) = C/HLT,N

OUTPUT:
COUNTL
COUNTR
COUNTLR
LSQP1
LSQP2
LSQP3
LSQP4
LSQP5
LSQP6
LSQSP1
LSQSP3
LSQDN

SUBROUTINES: Program - CHGCWD

STORAGE REQUIREMENTS: 18 Cells

DESCRIPTION:
COUNTL = C/HLT,N; C/HLT,0
COUNTR = C/HLT,0; C/HLT,N
COUNTLR = C/HLT,N; C/HLT,N

LSQP1 = C/HLT, TERMS + N; C/JMP, LSQMULT
LSQP2 = C/HLT, A11 + N^2 + 1; C/JMP, LSQLOD2
LSQP3 = C/HLT, B11 + N; C/JMP, LSQLOC
LSQP5 = C/HLT, N-1
LSQP6 = C/HLT, N + 1; C/HLT, N + 1

LSQSP1 = C/HLT, N
LSQSP3 = C/HLT,N; C/HLT,N
LSQDN = C/TDM, 0,2; C/AIXOL, N, 2

PURPOSE: If range or angles are observed, to compute the residuals and differential expressions which are entered in the least-squares matrix.

CALL SEQUENCE:

```
┌ TJM     CMPCFY   -   for range  ┐
{  or                              }
└ CM      CMPCFY   -   for angles ┘

  JMP     CMPCF
  TAM     (unweighted residual)
  TQM     (weighted residual)
```

INPUT: KNTRL = 0‾‾‾‾‾0XXXXXXX,

The last 7 bits correspond to the 7 elements

B, i, Ω, U_o, a_{yn}, a_{xn}, n.

If = 1, then the element is being corrected.

LTERMS = C/HLTR, TERMS

and output of COEFF2 subroutine

OUTPUT: See description.

SUBROUTINES: Program - CDLTB

STORAGE REQUIREMENTS: 41 Cells

DESCRIPTION: The following formulas are solved for, as indicated by KNTRL, and packed into the TERMS buffer to be entered in the least squares matrix by subroutine LSQR.

If ρ is observed, the terms are:

$$0,1 = \left[(U_N)\ (\underline{L}\cdot\underline{V}) + (R_N)\ (\underline{L}\cdot\underline{U}) \right] \frac{1}{\sigma_\rho} \qquad \text{for } n$$

$$1,1 = \left[(U_{XN})\ (\underline{L}\cdot\underline{V}) + (R_{XN})\ (\underline{L}\cdot\underline{U}) \right] \frac{1}{\sigma_\rho} \qquad \text{for } a_{xn}$$

$$2,1 = \left[(U_{YN})\ (\underline{L}\cdot\underline{V}) + (R_{YN})\ (\underline{L}\cdot\underline{U}) \right] \frac{1}{\sigma_\rho} \qquad \text{for } a_{yn}$$

$$3,1 = \left[(U_u)\ (\underline{L}\cdot\underline{V}) + (R_u)\ (\underline{L}\cdot\underline{U}) \right] \frac{1}{\sigma_\rho} \qquad \text{for } U_o$$

DESCRIPTION:
(continued)

$$4,1 = \left[(\cos i)\ (\underline{L} \cdot \underline{V}) - (\sin i)\ (\cos U)\ (\underline{L} \cdot \underline{W}) \right] \frac{r}{\sigma_\rho} \qquad \text{for } \Omega$$

$$5,1 = \left[(\sin U)\ (\underline{L} \cdot \underline{W}) \right] \frac{r}{\sigma_\rho} \qquad \text{for } i$$

6,1 = (See results of CDLTB subroutine; it is used to compute this term) for B

$$7,1 = (\rho - \rho_c) \frac{1}{\sigma_\rho} \qquad \text{weighted residual}$$

The unweighted residual is $(\rho - \rho_c)$.

If A or α is observed:

In the formulas for n through i, substitute $\underline{A}$ for $\underline{L}$, and $\frac{1}{\rho_c \sigma_A}$ for $\frac{1}{\sigma_\rho}$

The weighted residual is: $\rho_c\ (A \cdot \Delta L)\ \frac{1}{\rho_c \sigma_A}$;

The unweighted residual is: $\rho_a\ (A \cdot \Delta L)$

If h or α is observed:

In the formula for n through i, substitute $\underline{D}$ for $\underline{L}$, and $\frac{1}{\rho_c \sigma_h}$ for $\frac{1}{\sigma_\rho}$.

The weighted residual is: $\rho_c\ (\underline{D} \cdot \underline{\Delta L})\ \frac{1}{\rho_c \sigma_h}$

and the unweighted residual is $\rho_c\ (\underline{D} \cdot \Delta L)$.

PURPOSE: To control the differential correction.

CALL SEQUENCE:

```
JMP   CNTRL
JMP   (Error)
(Normal Return)
```

INPUT: Observations starting in EBLOC

OANDE = C/HLT, OBLOC; C/HLT, EBLOC

$$\left.\begin{array}{l}\text{XLO}\\ \text{AXO}\\ \text{AYO}\\ \text{AZO}\\ \text{HXO}\\ \text{HYO}\\ \text{HZO}\end{array}\right\}\ \text{Elements at epoch}$$

Input required for all subroutines used.

Input from P Card 3.

OUTPUT:

DLTNN = $\Delta n/_{n}$

DLTAX = Δa_{xn}

DLTAY = Δa_{y}

DLTUO = ΔU_{o}

DLTND = $\Delta\ \Omega$

DLTIN = $\Delta\ i$

DLTB = $\Delta B/_{B}$

RMS = root mean square of weighted residuals

SUBROUTINES:

System - PANT

Program - PAGECON, MOVDAT, INITIAL, CDLTBIN, SAV5PTS, ADBASH, REVSUB, SAVICK, SAVWBF, DIVDIF, SAVW, SETW, COEFF2, RESW, RESOUT1, SAVELM, COMRMS, LSQS

STORAGE REQUIREMENTS: 106 Cells

DESCRIPTION: This subroutine is primarily a logic routine to control use of the subroutines which are necessary for a differential correction

The observations are stored and retrieved in order starting at epoch. The closest observation before epoch is the first; then order the observations backward in time to the earliest one before epoch. The observations after epoch are stored next in chronological order Epoch can be anywhere in relation to the observations, i.e. before, after, or in the middle. The integration control is set to handle each situation most efficiently. The integration starts at epoch and goes back in time to the earliest observation; then everything is reinitialized and integration begins at epoch and goes forward in time to the latest observation.

If at any time during the integration the time or revolution number requested for the new epoch is found (P Card 3), the elements are saved to avoid repetition of integration. If the new epoch falls outside the span of the observations, subroutine PRERES will continue or initialize the integration to find the new epoch elements (See subroutine PRERES).

A. The following are executed only once:

(1) Set these locations $\neq 0$.

PRJ SAV · indicates new epoch elements have not been found

SAVELEM - subroutine SAVELM has not been called

FIRST - some observations may be before epoch

(2) Force a page for output

(3) Zero the least squares matrix buffers

DESCRIPTION:
(continued)

(4) Set the following:

REJCNT = 0/0 (REJCNT is number of rejected residuals)

RCNT = 0/0 (RCNT is number of accepted residuals)

SUM = F/0 (SUM is sum of squares of weighted residuals)

REV = EPREV (EPREV is epoch revolution)

(5) Retrieve the first observation. If no observations, take error exit. Otherwise, determine whether the first observation is before or after epoch, i.e. should the integration be backward or forward in time with epoch as a base? Set necessary switches accordingly.

B. Initialize for the first integration:

(1) Call subroutine INITIAL to start with the epoch elements.

(2) Initialize the CDLTB subroutine with CDLTBIN.

(3) Use subroutine SAV5PTS to save the epoch elements as the first entry in the interpolation buffer (ICK).

C. Begin the basic integration loop:

(1) Integrate for the next point using ADBASH subroutine.

(2) Update the revolution number with REVSUB subroutine.

(3) Save the element set with subroutine SAV5PTS.

(4) If the buffer for interpolation does not contain 5 elements sets, go to step C(1). Otherwise go to D.

DESCRIPTION:
(continued)

D. Look for new epoch elements:

(1) If the new epoch elements have been found, go to step E. Otherwise determine if searching for a revolution number or a time.

(2) If by revolution, test if current revolution number is equal to the new epoch revolution number. If not, go to E. If it is, save the last element set in the ICK buffer as the element set corresponding to the revolution number. Save other values necessary for PRERES subroutine and set PRDSAV = 0 indicating elements have been found.

(3) If by time, test if new epoch time is within the time span of the interpolation buffer. If not, go to E. Otherwise, interpolate for the elements at the new epoch time. Save other necessary values for PRERES subroutine and set PRDSAV = 0.

E. Search for elements corresponding to observation time:

(1) If the observation time is not within the time span of the interpolation buffer, go to step C to omit the first time and add a new one.

(2) If it is within the time span, interpolate for the elements at the observation time. Pass these elements to the COEFF2 subroutine which will compute the residuals and the partial derivatives and enter them in the least squares matrix buffers.

(3) Output the residuals if requested.

(4) Retrieve the next observation:
If on the same side of epoch as the previous observation, go to step E;
if not, go to step F.
If end of observations, go to step G.

DESCRIPTION:
(continued)

F. Reinitialize to integrate forward from epoch:

(1) If already initialized, go to step E. If not, test if new epoch elements are before epoch.

(a) If yes, and if elements have not been found, save the buffers necessary to restart integration by PRERES subroutine.

(b) If after epoch, reinitialize several switches and go to step B.

G. Test for new epoch elements:

(1) If the new epoch elements have been found, go to step H. If not, determine where new epoch is in relation to epoch. Save buffers to restart under several conditions. At times it is more convenient to initialize for the integration to the new epoch elements, i.e. if all the observations are before epoch and the new epoch is after epoch.

H. Solve for the delta elements:

(1) Compute the root mean square of the residuals.

(2) If there are enough observations accepted, solve the least squares matrix with subroutine LSQS. If not, take an error exit to location FINISH.

(3) The results of solving the least squares matrix (delta elements) are in a buffer. Unpack the buffer and exit.

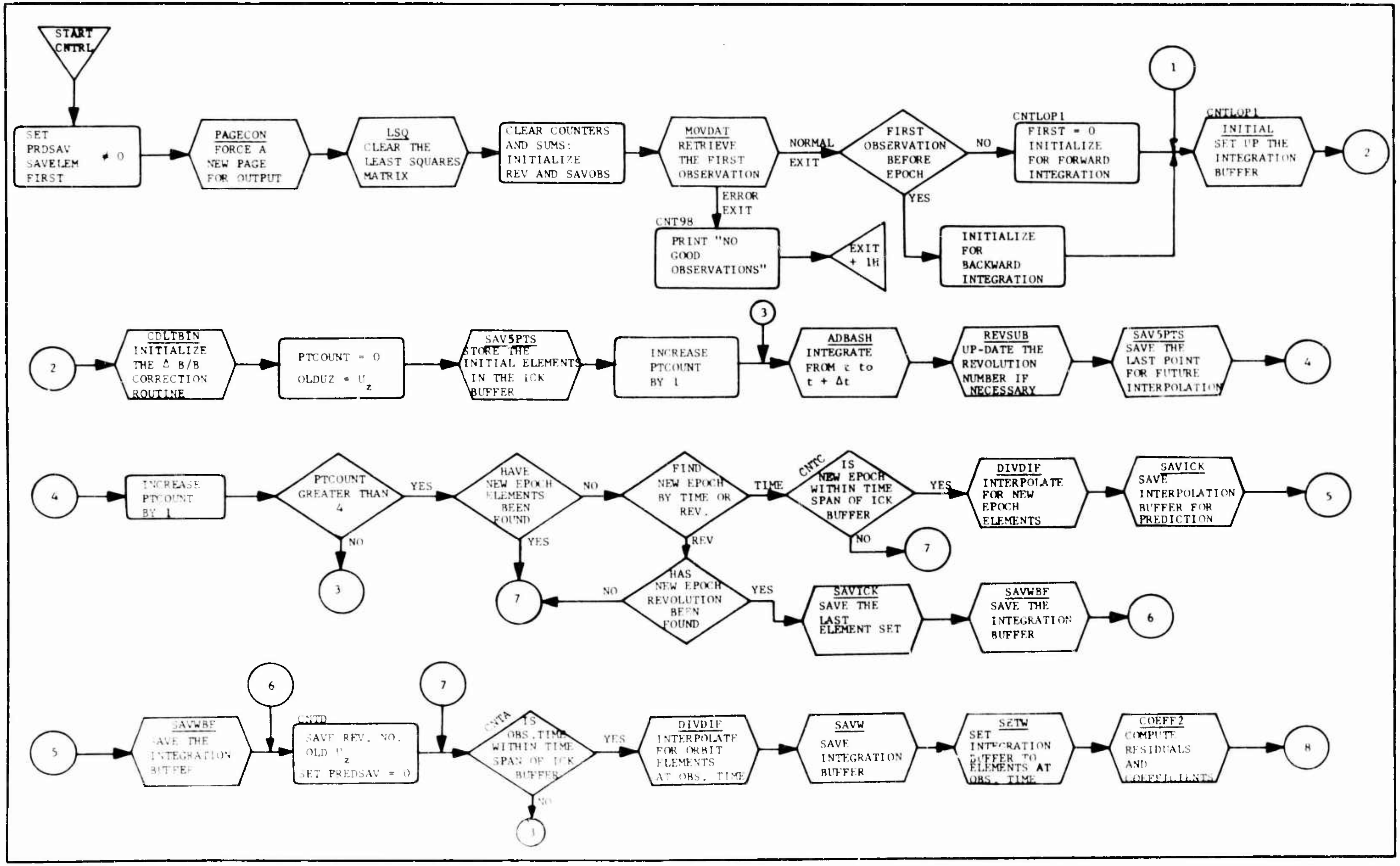


FIGURE [illegible]. CNTRL FLOW DIAGRAM (1 of 2)

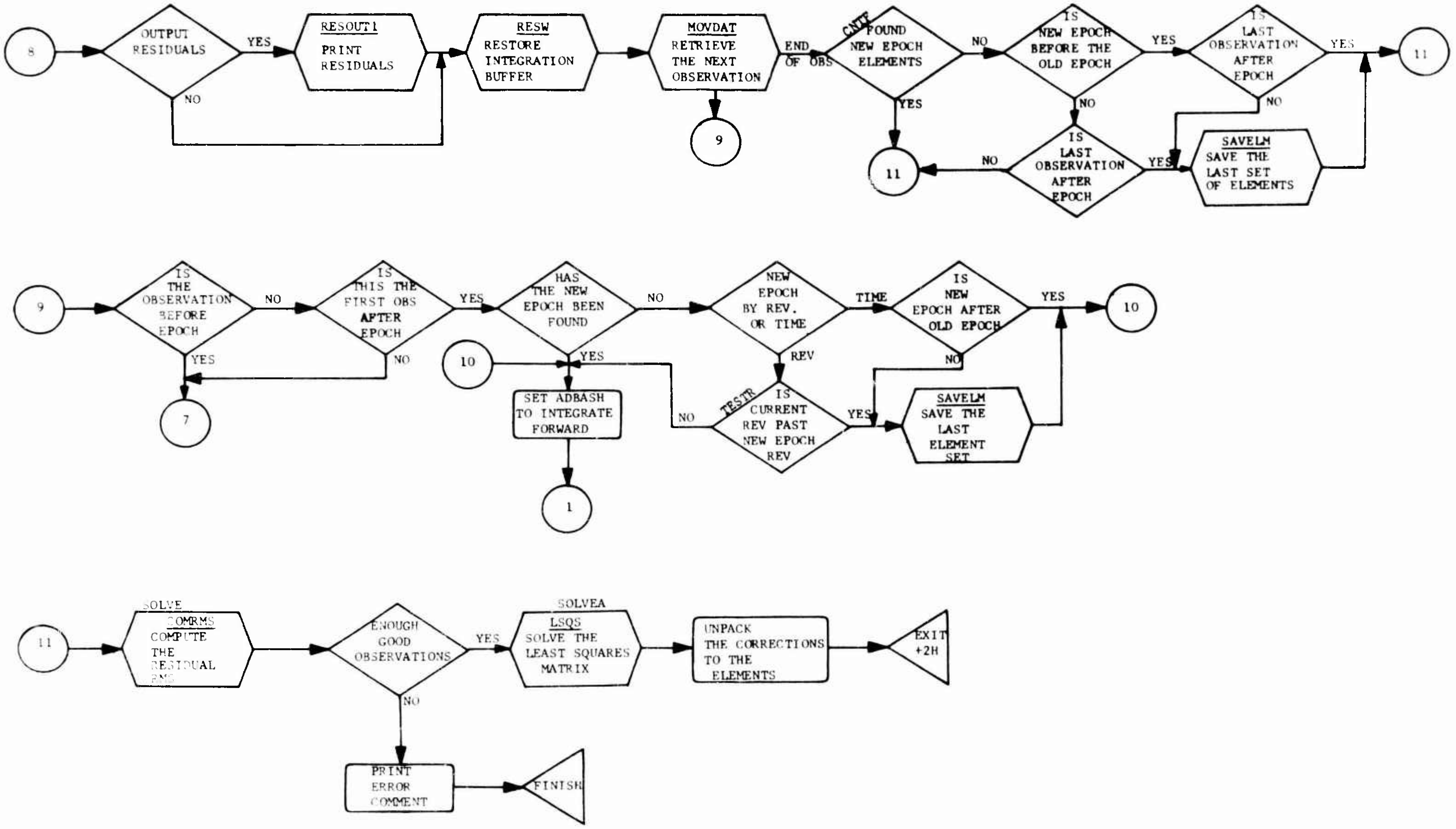


FIGURE 6. CNTRL FLOW DIAGRAM (2 of 2)

PURPOSE: To compute the coefficients for the least squares matrix and residuals for each observed quantity.

CALL SEQUENCE: JMP COEFF2

INPUT:

OBFLG = observed quantities

SIGMA1 = σ_ρ

SIGMA2 = $\sigma_{\dot{\rho}}$

SIGMA3 = σ_a

SIGMA4 = σ_h

OUTPUT:

RGSDL = $(\rho - \rho_c)$ (KM)

ARSDL = $\rho_c \, \underline{A} . \, (\underline{L} - \underline{L}_c)$ (KM)

DRSDL = $\rho_c \, \underline{D} . \, (\underline{L} - \underline{L}_c)$ (KM)

RRSDL = $(\dot{\rho} - \dot{\rho}_c)$ (KM/SEC)

RGFLG =
ALFLG =
DLFLG =
RRFLG =
} Δ if accepted
} * if rejected

REJFLG = * if any rejected
Δ if all accepted

Modified A and B matrices in A11 and B11 buffers.

SUBROUTINES: Program - SUBXYZ, RHOSB, COMXLX, DOTPR, CMPCF, RESREJ1, LSQR, COMDEL, CORRD, REJECT1, REJECT2.

STORAGE REQUIREMENTS: 93 Cells

DESCRIPTION: First computes R's and U's at time t. Then tests OBFLG for observed quantities and computes the specified residuals and coefficients using subroutines CMPCF and CORRD. Tests if the residuals are accepted or rejected. If accepted, the coefficients are added to the matrices A and B using subroutine LSQR. If rejected, a flag is set, corresponding to the residual to indicate rejection.

DESCRIPTION (continued)

4 switches must be preset before calling this subroutine:

(1) for 1st pass only.

```
COEFR  JMP  REJECT1
COEFA  JMP  REJECT1
COEFH  JMP  REJECT1
COEFRR JMP  REJECT2
```

(2) for the 2nd through n passes:

```
COEFR  JMP  RESREJ1
COEFA  JMP  RESREJ1
COEFH  JMP  RESREJ1
COEFRR JMP  RESREJ1
```

COELTS
SPIRSEC

PURPOSE: To prepare values for output by ELMOUT.

CALL SEQUENCE: TMP COELTS

INPUT:

XLO
AXNO
AYNO
HXO
HYO
HZO
B
REV
QO
SINI
ESQ
P
P3HA02
XNO

OUTPUT:

LPRINT = L_o (deg)
AXPRINT = a_{xn}
AYPRINT = a_{yn}
HXPRINT = h_x
HYPRINT = h_y
HZPRINT = h_z
BPRINT = B
PREV = Rev No.
HSUBQP = h_q (km)
PAPRINT = p_A (minutes)

SUBROUTINES: (None)

STORAGE REQUIREMENTS: 19 Cells

DESCRIPTION: Computes HSUBQP and PAPRINT - moves other values to output cells. Used to prepare initial, final, and prediction elements for output.

PURPOSE:	To compute ΔL_x, ΔL_y, ΔL_z
CALL SEQUENCE:	JMP COMDEL
INPUT:	XLSUPX XLSUBY from AZREC or ALREC subroutines. XLSUBZ XLX = ρ_x / ρ_c XLY = ρ_y / ρ_c XLZ = ρ_z / ρ_c
OUTPUT:	DELTX DELTY DELTZ
SUBROUTINES:	(None)
STORAGE REQUIREMENTS:	6 Cells
DESCRIPTION:	DELTX = XLSUBX - XLX DELTY = XLSUBY - XLY DELTZ = XLSUBZ - XLZ

COMDQ
SPIRDEC

PURPOSE: To compute delta q.

CALL SEQUENCE: JMP COMDQ

INPUT:

DLTNN = $\Delta n/n$

XNO = n_o

DLTAX = Δa_x

DLTAY = Δa_y

AXNO = a_{xno}

AYNO = a_{yno}

QO = q_o

EO = e_o

AO = a_o

OUTPUT: DELTAQ = Δq

SUBROUTINES: Philco - FSQRT

STORAGE REQUIREMENTS: 13 Cells

DESCRIPTION: If $e_o \neq 0$, then

$$q = - 2/3 \, \Delta n_{/n} \, q_o - a_o \left(\frac{a_{xno} \, \Delta a_{xn} + a_{yno} \, \Delta a_{yn}}{e_o} \right)$$

If $e_o = 0$, then

$$q = - 2/3 \, \Delta n_{/n} \, q_o - a_o \sqrt{\Delta a_{xn}^2 + \Delta a_{yn}^2}$$

PURPOSE:	To compute root mean square
CALL SEQUENCE:	JMP COMRMS
INPUT:	RCNT = C/HLT,N where N = number of accepted weighted residuals SUM = sum of squares of accepted weighted residuals
OUTPUT:	ACCCNT = C/HLT,N RCNT = N (floating point) RMS = root mean square
SUBROUTINES:	(None)
STORAGE REQUIREMENTS:	16 Cells
DESCRIPTION:	ACCCNT = RCNT RCNT is converted to F.P. RMS = $\sqrt{\text{SUM/RCNT}}$

PURPOSE:	To compute $\underline{L}$
CALL SEQUENCE:	JMP COMXLX
INPUT:	RHOX = ρ_x RHOY = ρ_y RHOZ = ρ_z RHOC = ρ_c
OUTPUT:	XLX = L_x XLY = L_y XLZ = L_z
SUBROUTINES:	(None)
STORAGE REQUIREMENTS:	5 Cells
DESCRIPTION:	XLX = ρ_x / ρ_c XLY = ρ_y / ρ_c XLZ = ρ_z / ρ_c

PURPOSE: To compute coefficients of range-rate correction equations and the range-rate residuals.

CALL SEQUENCE:

```
JMP  CORRD
TAM  (unweighted residual)
TQM  (weighted residual)
```

INPUT:

KNTRL = 0 0XXXXXXX

LTERMS = C/HLTR, TERMS

and output of COEFF2 subroutine

Switch CORRDS4 must be preset to JMP CORRDC before entering this routine.

OUTPUT: See description.

SUBROUTINES: Program - RDTSB, CDLTBA

STORAGE REQUIREMENTS: 15 Cells

DESCRIPTION: First these formulas must be solved:

EXOM = $a\ (e \cos E - e^2)$

EYOM = $\sqrt{1 - e^2}\ (a\ e \cos E)$

VDOT = $r\dot{v}/r$

RSQ = r^2

RCUBE = r^3

RVDOT = $r\dot{v}$

XMUA32 = $\sqrt{\mu}\ a^{3/2}$

RDOTU = $\dfrac{\sqrt{\mu}\ a^{3/2}}{r^3}\left[a\ (e \cos E - e^2)\right]$

UDOTU = $\dfrac{\sqrt{\mu}\ a^{3/2}}{r^3}\ \sqrt{1-e^2}\ (a\ e \cos E)$

DESCRIPTION: (continued)

$$RDOTN = \dot{r}/3 + (U-U_o)\ (RDOTU)$$

$$UDOTN = r\dot{v}/3 + (U-U_o)\ (UDOTU)$$

$$RDTXN = \left[\sin\ (E+\omega) - a_{yn} - (a_{xn})\ (e\ \sin\ e)\right]\frac{\sqrt{\mu}\ a^{5/2}}{r^3}$$

$$RDTYN = \left[a_{xn} - \cos\ (E+\omega) - (a_{yn})\ (e\ \sin\ e)\right]\frac{\sqrt{\mu}\ a^{5/2}}{r^3}$$

$$UDTXN = \frac{\sqrt{1-e^2}\sqrt{\mu}\ a^{5/2}}{r^3}\left[\cos\ (E+\omega) - a_{xn}\ \left(\frac{r^2}{pa}+1\right)\right]$$

$$UDTYN = \frac{\sqrt{1-e^2}\sqrt{\mu}\ a^{5/2}}{r^3}\left[\sin\ (E+\omega) - a_{yn}\ \left(\frac{r^2}{pa}+1\right)\right]$$

$$RDOTOR = \dot{r}/r$$

$$TEMP1 = \dot{\rho}_x U_x + \dot{\rho}_y U_y + \dot{\rho}_z U_z$$

$$TEMP2 = \dot{\rho}_x V_x + \dot{\rho}_y V_y + \dot{\rho}_z V_z$$

$$TEMP3 = \dot{\rho}_x W_x + \dot{\rho}_y W_y + \dot{\rho}_z W_z$$

Having computed these formulas, the following quantities are stored in the TERMS buffer as described by the bits in KNTRL:

$$\left.\begin{array}{l} C_{\Delta n/n} \\ C_{\Delta a_{xn}} \\ C_{\Delta a_{yn}} \\ C_{\Delta U_o} \\ C_{\Delta \Omega} \\ C_{\Delta i} \\ C_{\Delta B/B} \end{array}\right\} \text{Correction coefficients}$$

R (weighted residual)

PURPOSE: To apply corrections to the elements.

CALL SEQUENCE: JMP CREKT

INPUT:

DLTNN = $\Delta n/n$

DLTAX = Δaxn

DLTAY = Δayn

DLTND = Δ

DLTIN = Δi

DLTUO = ΔU_o

DLTB = $\Delta B/B$

OUTPUT: See description

SUBROUTINES: Philco - FSIN, FCOS, FSQRT, FLOG2X, F2X

STORAGE REQUIREMENTS: 46 Cells

DESCRIPTION: The delta elements are applied and new elements are computed as follows:

XNO = $n_o = n_o \ (1 + \Delta n/n)$

B = $C_d \ A/m = B \ (1 + \Delta B/B) \quad |\Delta B/B| \le 0.25$

UO = $U_o = U_o + \Delta U_o$

AXNO = $a_{xno} = a_{xno} + \Delta a_{xno}$

AYNO = $a_{yno} = a_{yno} + \Delta a_{yno}$

ESQ = $e_o^2 = a_{xno}^2 + a_{yno}^2$

EO = $e_o = \sqrt{e_o^2}$

XNODEO = $\Omega_o = \Omega_o + \Delta\Omega_o$

SINO = $\sin \Omega_o$

COSO = $\cos \Omega_o$

DESCRIPTION:
(continued)

XINCL = $i = i + \Delta i$

COSI = $\cos i$

SINI = $\sin i$

WX = $(\sin i)\ (\sin \Omega)$

WY = $-(\sin i)\ (\cos \Omega)$

WZ = $\cos i$

XLO = $L_o = U_o + \Omega$

AO = $a_o = \left(\frac{k_e}{n_o}\right)^{2/3}$

P = $p = a_o\ (1 - e_o^2)$

RTP = $\sqrt{p}$

HXO = $h_{xo} = \sqrt{p}\ W_x$

HYO = $h_{yo} = \sqrt{p}\ W_y$

HZO = $h_{zo} = \sqrt{p}\ W_z$

AXO = $a_{xo} = (\cos \Omega)\ (a_{xno}) - (\cos i)\ (\sin \Omega)\ (a_{yno})$

AYO = $a_{yo} = (\sin \Omega)\ (a_{xno}) + (\cos i)\ (\cos \Omega)\ (a_{yno})$

AZO = $a_{zo} = (\sin i)\ (a_{yno})$

QO = $q_o = a_o\ (1 - e_o)$

DELOUT1
SP1RDEC

PURPOSE: To print the delta elements after each D. C. pass.

CALL SEQUENCE: JMP DELOUT1

INPUT:

DLTNN = $\Delta n/n$
DLTAX = Δa_{xn}
DLTAY = Δa_{yn}
DLTUO = ΔU_o
DLTNO = $\Delta \Omega$
DLTIN = Δi
DLTB = $\Delta B/B$
all floating point

RMS = weighted root mean square

OUTPUT:

DELTA N/N
DELTA AXN
DELTA AYN
DELTA UO
DELTA NODE
DELTA I
DELTA B
RMS

SUBOUTINES:
System - GLOP, PANT
Program - PAGECON

STORAGE REQUIREMENTS: 53 Cells

DESCRIPTION: Forces a page, prints headings, values, 10 spaces, and updates PAGECON.

PURPOSE:	To print the last pass element corrections, old and new rms, and accepted and rejected count.
CALL SEQUENCE:	JMP DELOUT2
INPUT:	DLTNN DLTAX DLTAY DLTJO DLTND DLTIN DLTB RMS ORMS ACCCNT REJCNT
OUTPUT:	DELTA N/N DELTA NXN DELTA AYN DELTA UO DELTA NODE DELTA I DELTA B OLD RMS NEW RMS No. of residuals used and rejected
SUBROUTINE:	System - PANT, GLOP Program - PAGECON
STORAGE REQUIREMENTS:	56 Cells
DESCRIPTION:	Forces a page. Prints the comment "DC Converged - The next corrections would be...." Then prints headings, delta elements, old and new rms, accepted and rejected residual count, gives 8 spaces and updates PAGECON.

DELTAU
SPIRDEC

PURPOSE:	To compute Δu
CALL SEQUENCE:	JMP DELTAU
INPUT:	XOBS YOBS ZOBS UX UY UZ
OUTPUT:	DELU
SUBROUTINES:	Program - ARCTAN
STORAGE REQUIREMENTS:	19 Cells
DESCRIPTION:	RU = XOBS·UX + YOBS·UX + ZOBS·UZ = COSDU SINDU = WX (UY ZOBS - UZ·YOBS) + WY (UZ·XOBS - UX·ZOBS) + WZ (UX·YOBS - UY·XOBS) DELU = $\tan^{-1}$ (SINDU/RU) (mod 2π)

PURPOSE: To compute the derivatives of the $\underline{N}$, $\underline{M}$ elements with respect to time, using the effects of radiation pressure, drag and/or bulge perturbation as specified on P Cards.

CALL SEQUENCE: JMP DERIV

INPUT: See input for MARTINI, JNDRAG, RDPRES and SUBXYZ subroutines.

OUTPUT:

XDGR = $\dot{x}'$
YDGR = $\dot{y}'$
ZDGR = $\dot{z}'$
DGR = D'
DDGR = $\dot{D}'$
D = D
AXGR = a'_x
AZGR = a'_z
SMLGR = ℓ'

W + 187 = d(L)/dt
188 = $d(a_x)/dt$
189 = $d(a_y)/dt$
190 = $d(a_z)/dt$
191 = $d(h_x)/dt$
192 = $d(h_y)/dt$
193 = $d(h_z)/dt$

SUBROUTINES: Program - SUBXYZ, MARTINI, JNDRAG, RDPRES

STORAGE REQUIREMENTS: 97 Cells

DESCRIPTION: Uses SUBXYZ to compute $\underline{r}$, $\dot{\underline{r}}$; computes specified perturbative effects to be used; and computes the derivatives, storing them in the ADBASH buffer (W + 187 to W + 193).

PURPOSE: To interpolate for elements using a fourth-order divided difference method.

CALL SEQUENCE: JMP DIVDIF

INPUT: ICK buffer (See description)

T = t (min. since epoch)

TBUF = temporary storage buffer

OUTPUT:

$$\left.\begin{array}{rcl} ICK + 40 & = & L_o \\ 41 & = & a_x \\ 42 & = & a_y \\ 43 & = & a_z \\ 44 & = & h_x \\ 45 & = & h_y \\ 46 & = & h_z \end{array}\right\} \text{at time } t$$

SUBROUTINE: None

STORAGE REQUIREMENTS: 38 Cells

DESCRIPTION: Given the 5 sets of time and elements in buffer ICK, the subroutine will interpolate for the elements at time t, where $t_o \leq t \leq t$ 4, using the following formula to interpolate for each element:

$$x = x_o + (t-t_o)\left[\frac{x_1-x_o}{t_1-t_o}\right] + (t-t_o)(t-t_1)\left[\frac{\left(\frac{x_2-x_1}{t_2-t_1}\right) - \left(\frac{x_1-x_o}{t_1-t_o}\right)}{t_2-t_o}\right]$$

$$+ (t-t_o)(t-t_1)(t-t_2)\left[\frac{\left(\frac{\frac{x_3-x_2}{t_3-t_2} - \frac{x_2-x_1}{t_2-t_1}}{t_3-t_1}\right) - \left(\frac{\frac{x_2-x_1}{t_2-t_1} - \frac{x_1-x_o}{t_1-t_o}}{t_2-t_o}\right)}{t_3-t_o}\right]$$

DESCRIPTION:
(continued)

$$+ (t-t_o)(t-t_1)(t-t_2)(t-t_3)\left[\frac{\left(\frac{\frac{x_4-x_3}{t_4-t_3} - \frac{x_3-x_2}{t_3-t_2}}{t_4 - t_2}\right) - \left(\frac{\frac{x_3-x_2}{t_3-t_2} - \frac{x_2-x_1}{t_2-t_1}}{t_3 - t_1}\right)}{(t_4-t_1)\quad(t_4-t_o)}\right.$$

$$\left. - \frac{\left(\frac{\frac{x_3-x_2}{t_3-t_2} - \frac{x_2-x_1}{t_2-t_1}}{t_3-t_1}\right) - \left(\frac{\frac{x_2-x_1}{t_2-t_1} - \frac{x_1-x_o}{t_1-t_o}}{t_2-t_o}\right)}{(t_3-t_o)\quad(t_4-t_o)}\right]$$

ICK Buffer Format

ICK + 0	t_o	5 sets of time and elements
+ 1	L_o	
+ 2	a_x	
+ 3	a_y	
+ 4	a_z	
+ 5	h_x	
+ 6	h_y	
+ 7	h_z	
⋮		
+ 32	t_4	
+ 33	L_o	
+ 34	a_x	
+ 35	a_y	
+ 36	a_z	
+ 37	h_x	
+ 38	h_y	
+ 39	h_z	

PURPOSE: To compute dot products of vector in the (A), (Q), (D) registers with $\underline{U}$, $\underline{V}$, $\underline{W}$

CALL SEQUENCE:
TMA (L_x)
TMQ (L_y)
TMD (L_z)
JMP DOTPR

INPUT:
UX
UY
UZ
VX
VY
VZ
WX
WY
WZ

OUTPUT:
ADOTU
ADOTV
ADOTW

SUBROUTINES: (None)

DESCRIPTION:

$$\mathrm{ADOTU} = L_x U_x + L_y U_y + L_z U_z$$

$$\mathrm{ADOTV} = L_x V_x + L_y V_y + L_z V_z$$

$$\mathrm{ADOTW} = L_x W_x + L_y W_y + L_z W_z$$

PURPOSE:	To output initial, final, and new epoch elements at the end of the differential correction.
CALL SEQUENCE:	JMP ELMOUT
INPUT:	Initial elements in INELT buffer. New epoch elements in PREDBF buffer Final elements in: XLO AXNO AYNO HXO HYO HZO B REV
OUTPUT:	See description
SUBROUTINES:	Program - RINEL, BCDTIM, COELTS, PRERES, PAGECON System - PANT, GLOP
STORAGE REQUIREMENTS:	157 Cells
DESCRIPTION:	Prints comment: "Initial, Final, and New Epoch Elements" Prints headings, then restores initial elements to output cells and prints them. Prepares and prints final elements. Uses subroutine PRERES to obtain new epoch elements and prints them. Then prints comment: "End of DC" and updates PAGECON. Also prints card images of P Cards necessary to run CALIB. Punches the following cards in P Card formats: (1) Corrected elements (P1 & P2) (2) New Epoch elements (P1 & P2) (3) Cards for calibration program (P1, P2, P3) and the cards P5-P8 which were used in the current SPIRDEC run.

PURPOSE: To print the corrected elements after each differential correction.

CALL SEQUENCE: JMP ELMOUT1

INPUT:

PREV = rev. no.
LPRINT = L (deg)
AXPRINT = a_{xno}
AYPRINT = a_{yno}
HXPRINT = h_x
HYPRINT = h_y
HZPRINT = h_z
BPRINT = C_d A/m
HSUBQP = Hq (km)
PAPRINT = Period (min)

OUTPUT:

$$\left.\begin{array}{l}\text{OBYEAR}\\ \text{OBMO}\\ \text{OBDAY}\\ \text{FOBHR}\\ \text{FOBHR}\\ \text{FOBMIN}\\ \text{FOBSEC}\end{array}\right\}\ \text{BCD Time}$$

$$\left.\begin{array}{l}\text{REV}\\ \text{TIME}\\ \text{L}\\ \text{AXN}\\ \text{AYN}\\ \text{HXO}\\ \text{HYO}\\ \text{HZO}\\ \text{B}\\ \text{PER. ALT.}\\ \text{PA}\end{array}\right\}\ \begin{array}{l}\text{These are not program locations}\\ \text{but output headings}\end{array}$$

SUBROUTINES: Program - COELTS, PAGECON, BCDTIM, PRNTMAT
System - PANT, GLOP

STORAGE REQUIREMENTS: 53 Cells

DESCRIPTION: Uses BCDTIM to prepare time for output. Uses COELTS to compute values and correct elements to the output units. Prints "Corrected Elements". Then prints headings, values, and updates PAGECON. If the DC is converging, PRNTMAT will be called to print the standard deviations and correlation matrix of the delta elements. If divergent, the last corrected element set will be punched on P cards 1 and 2.

PURPOSE:	To wrapup the binary ephemeris .
CALL SEQUENCE:	JMP FTAPEW
INPUT:	TAPBUF = ephemeris buffer TAPCNT = C/HLT, TAPBUF + X
OUTPUT:	Sentinal of Z's
SUBROUTINES:	System - SYS, SYSNO, SYSIO
STORAGE REQUIREMENTS:	9 Cells
DESCRIPTION:	Writes the final block on the binary ephemeris tape on logical 10 - the sentinel being Z's in the current block. Then the tape is rewound with lockout.

PURPOSE: To retrieve sensor information.

CALL SEQUENCE:
JMP GETSEN
(No sensor data)
(Normal return)

INPUT:
BIASAD = C/HLT, EBLOC, C/HLT, BIBUF
BIBUF = SBUF = Modified buffer of sensor information
STAID = 00000SSS

OUTPUT:
PHIRD = φ
XLAMBA = λ
XOVCT = $x / \cos \theta$
CAPZ = Z

SUBROUTINES: None

STORAGE REQUIREMENTS: 10½ Cells

DESCRIPTION: Given a sensor number in STAID, the routine will search through SBUF until a match or Z's are found.

(1) If Z's are found, the error exit (+1H) is taken.

(2) If a match is found, the information will be unpacked to the cells listed under output.

PURPOSE: To retrieve weights.

CALL SEQUENCE:
JMP GETWGT
JMP (No Weights)
(Normal Return)

INPUT:
Weights in WBUF
WANDBI = C/HLT, WBUF; C/HLT, BIBUF
STAID = 00000SSS where SSS is the station number.

OUTPUT:
SIGMA1 = σ_ρ
SIGMA2 = $\sigma_{\dot{\rho}}$
SIGMA3 = σ_A
SIGMA4 = σ_h

SUBROUTINES: None

STORAGE REQUIREMENTS: 11 Cells

DESCRIPTION: Search WBUF for a match to STAID. If a match is found, unpack the 4 weights (noted under Output) and exit +2H. If no match is found, exit +1H.

GIPAR
SPIRDEC

PURPOSE: To prepare input for the GIPAR program.

CALL SEQUENCE: JMP GIPAR

INPUT:

T = minutes since epoch
ORGDAY = days since beginning of year
ORGTM = fraction of epoch day
X = x (in km or e.r.)
Y = y (in km or e.r.)
Z = z (in km or e.r.)
FTFLAG
PFLAG
GIPADR = left address is the next location in the GIPAR buffer

OUTPUT: An entry in the GIPAR buffer:

Word	0	x
	1	y
	2	z
	3	day number
	4	fraction of day
	5	ZZZZZZZZ

SUBROUTINES: Program - SEPSUB

STORAGE REQUIREMENTS: 21 Cells

DESCRIPTION: Before the first entry to this subroutine GIPADR must be set to C/HLT, EBLOC + 20 and EBLOC + 20 must be set = ZZZZZZZZ. The subroutine will update GIPADR as it makes entries to the buffer.

(1) Convert time from minutes since epoch to days since the beginning of the year and fraction of day; then store.

(2) Test PFLAG:

(a) If PFLAG = 0, store x, y, z

(b) If PFLAG ≠ 0, convert x, y, z from km. to e.r. and store. PFLAG ≠ 0 means subroutine SUBOUT has converted x, y, z to km.

(3) Store sentinel of Z's and update GIPADR.

HEAD
SPIRDEC

PURPOSE:	To output a new page with the page headings.
CALL SEQUENCE:	JMP HEAD
INPUT:	Parameters set by 1HEAD1 subroutine.
OUTPUT:	New page with the described page headings.
SUBROUTINES:	System - PANT, GLOP
STORAGE REQUIREMENTS:	52 Cells
DESCRIPTION:	1st line - Spiral Decay SPDC Program Page X 2nd line - Satellite No. = XXX Satellite name - x-x(10 charact 2nd line (cont) - Element set no. = xxx Time of EPOCH = YY MM DD HH MM SS.SSS

PURPOSE:	To initialize the page heading routine (HEAD)
CALL SEQUENCE:	JMP IHEAD1 (Return)
INPUT:	SATN = 0 — 0 NNN, Satellite number ORGDA = days since 1950 } Epoch ORGTM = fraction of day }
OUTPUT:	PAGENO = 0 — 0 PDAY = MMO — 0 PYEAR = , Δ 19YY00 SATEL = NNNO — 0, N = SATN HEAD2 + 3 H = ADDR of BCD MONTH HEAD5 to HEAD5 + 5 = (BCD EPOCH Time in cells for output call sequence.)
SUBROUTINES:	System - AKLOK Program - BCDTIM
STORAGE REQUIREMENTS:	26 Cells
DESCRIPTION:	Uses AKLOK to get PDAY and PYEAR - Uses BCDTIM to get Epoch in BCD ready for output Sets up parameters necessary to output page heading as done by HEAD.

PURPOSE:	To initialize for the ADBASH subroutine.
CALL SEQUENCE:	JMP INITIAL
INPUT:	XLO AXO AYO AZO HXO HYO HZO
OUTPUT:	See description.
SUBROUTINES:	Program - WSETUP
STORAGE REQUIREMENTS:	9 Cells
DESCRIPTION:	Moves elements to W buffer by calling WSETUP Sets I (W + 3) = F/O Sets switch ADP to 1/1T15 Sets switches ADDER1 to ADDER7 to a (JMP DERIV) in the ADBASH subroutine.

INITL
SPIRDEC

PURPOSE:	To print the cards images of the parameter cards (P cards 1 thru 10) and process them.
CALL SEQUENCE:	JMP INITL JMP (ERROR) (NORMAL)
INPUT:	Parameter cards (P cards) in CONBUF
OUTPUT:	See description.
SUBROUTINES:	Program - PAGECON, INPUT System - PANT
STORAGE REQUIREMENTS:	19 Cells
DESCRIPTION:	Prints card images of cards in CONBUF; then calls INPUT to process these cards.

PURPOSE: To unpack and validate the input parameter cards (1-10).

CALL SEQUENCE:

```
JMP  INPUT
JMP  (error)
(Normal return)
```

INPUT:

P Cards (1-10) in CONBUF
PCOUNT = C/HLT, N
    where N = number of P cards

OUTPUT: See description.

SUBROUTINES:

Program - XSRCH, RR2AHL, CARDER, SAVCON
System - FSKLOK, FXFLT, INITEL, NXTELM, FYKLOK

STORAGE REQUIREMENTS: 370 Cells

DESCRIPTION: The input parameter cards need not be in order; they are unpacked according to the number in Column 79, using subroutine XSRCH.

If no error is found on a card, a bit will be set in location CARDS:

1/1T0 - P Card 1
1/1T1 - P Card 2
1/1T2 - P Card 3
1/1T3 - P Card 4
1/1T4 - P Card 5
1/1T6 - P Card 6
1/1T7 - P Card 7
1/1T8 - P Card 8
1/1T9 - P Card 9
1/1T10 - P Card 10

DESCRIPTION:
(continued)

The remainder of any card in error is not validated beyond the error; however, an attempt will be made to validate the remaining cards. Subroutine CARDER will be used to output the card image of the erroneous card with a comment indicating the first field in error; location CARDSW will be set ≠ 1. If an error occurs during XSRCH conversion, a comment will be printed with the card image, but with no field indicated.

The subroutine XSRCH will unpack each of the cards into the locations:

All floating point

P Card 1	If Col. 78 = A		If Col. 78 = B	
Col. 1 - 12	XLO	(rad)	X	(km)
13 - 24	AXNO		Y	(km)
25 - 36	AYNO		Z	(km)
37 - 49	HXO	(er/kemin)	XDOT	(m/sec)
50 - 62	HYO	(er/kemin)	YDOT	(m/sec)
63 - 75	HZO	(er/kemin)	ZDOT	(m/sec)

If type B elements are input, subroutine RR2AHL will be used to convert them to type A elements with the results stored in the locations for type A elements. If no errors set bit 0 in CARDS = 1

P Card 2

Col. 1 - 5	SATN	(Binary)
6 - 15	SATNM, SATNM+1	(Binary)
17 - 32	YY, YY + 1	(Binary)
33 - 37	EPREV	(F.P.)
38 - 40	ELNO	(FX.P)

Having unpacked the card, the following will be done:

TOY = 0———0 YY, YY = BCD year

Using subroutine FSKLOK;

ORDGA = days since 1950 to epoch (F.P.)

ORGTM = fraction of epoch day (F.P.)

Then the card image will be saved using subroutine SAVCON, for use by subroutine ELMOUT.

If no errors, set bit 1 in CARDS = 1

DESCRIPTION: (continued) <u>P Card 3</u>

Col.	1	CRAIG	(FX.P.)
	2	WGTFLG	(FX.P.)
	3	PW	(FX.P.)
	6	BFLAG	(FX.P.)
	7	DFLAG	(FX.P.)
	8	RPFLAG	(FX.P.)
	9	PREDFLG	(FX.P.)
	10-25	PRTIM, PRTIM+1	(BCD)
	26-32	SKNTRL	(BCD)
	33	RPT	(FX.P.)
	34	CNFLAG	(FX.P.)
	35	DQFLAG	(FX.P.)
	36-39	DQN	(F.P.)
	40-42	ABSMX	(F.P.)
	43-45	ABMX2	(F.P.)
	46-48	XISTSG	(F.P.)
	49-51	KAPPA	(F.P.)
	52-58	B	(F.P.)
	59	ROFLAG	(FX.P.)
	60	RESOPT	(FX.P.)
	62-64	CONTEST	(F.P.)
	66-68	F10	(F.P.)
	69-71	F10AV	(F.P.)
	72-75	AP	(F.P.)
	76-78	GAMMA	(F.P.)

DESCRIPTION: (continued) Having unpacked P Card 3, the following validations and conversions will be made:

If ROFLAG ≠ 0,1,2, set ROFLAG = 0

If RPT = 0, then ERROR (No. of iterations not specified)

If SKNTRL = 0, then ERROR (Elements to correct not specified)

If B is to be corrected:

1) If B = 0, then ERROR

If B is not to be corrected:

2) If DFLAG ≠ 0, then B, F10, F10AV, AP cannot = 0

3) If DFLAG = 0, then B, F10, F10AV, AP are not checked

If DQFLAG ≠ 0, then DQN cannot = 0

$$\left.\begin{array}{r}\text{If ABSMX} = 0\\ \text{and/or ABMX2} = 0\end{array}\right\}$$ then ERROR (Rejection criteria cannot = 0)

If XISTSG = 0, then set = F/1.5

If KAPPA = 0, then set = F/1

If RPFLAG ≠ 0, then GAMMA ≠ 0, and compute RPCON3 = GAMMA B/2.2

If DFLAG = ι, set RHO = F/0

If PREDFLG:

1) = 0, use FSKLOK to convert PRTIM, PRTIM+1 to days since 1950 and fraction of day respectively.

2) = 1, use FXFLT to convert PRTIM, PRTIM+1 to revolution number and store in PRTIM

3) = 2, do not use PRTIM, PRTIM+1

If CONTEST = F/0, set = F/.01

DESCRIPTION:
(continued)

Convert SKNTRL from BCD to binary in reverse order:

SKNTRL = 0XXXXXXX (N, AXN, AYN, U, Ω, i, B) BCD

to SKNTRL = 0———0XXXXXXX (B, i , Ω, U, AYN, AXN, N) binary

Convert: DQN from km to e.r.

ABSMX from km to e.r.

ABMX2 from km/sec to er/kemin

If no errors are found, set DCFLAG ≠ 0 and bit 2 = 1 in CARDS.

Use SAVCON to same card image for ELMOUT.

P Card 4

Col.	2-5	LTBUF+1	(F.P.)
	6	LTBUF+2	(FX.P.)
	7	LTBUF+3	(FX.P.)
	8	LTBUF+4	(FX.P.)
	9	FTFLAG	(FX.P.)
	10-25	PRTIME, PRTIME+1	(BCD)
	27	PFLAG	(FX.P.)
	41	BTFLAG	(FX.P.)
	42-45	LTBUF+5	(F.P.)
	49-51	PKAPPA	(F.P.)
	52-58	PB	(F.P.)
	66-68	PF10	(F.P.)
	69-71	PF10AV	(F.P.)
	72-75	PAP	(F.P.)
	76-78	PGAMMA	(F.P.)

DESCRIPTION: (continued)

The following validations and conversions will be made:

If PKAPPA = F/0, set = F/1

If DFLAG ≠ 0, then PB, PF10, PF10AV, PAP cannot = 0 or ERROR

If PFLAG = 0 and BTFLAG = 0, then ERROR (no output requested)

If BTFLAG = 1, then LTBUF + 5 cannot ≠ 0, or ERROR

If FTFLAG:

1) = 0, use FSKLOK to convert (BCD) PRTIME, PRTIME+1 respectively to days since 1950 and fraction of day.

2) = 1, use FXFLT to convert PRTIME, PRTIME+1 to revolution number and store in SREVF.

If LTBUF+4 ≠ 0, then PGAMMA cannot = 0, or ERROR

If no errors were found, set PPPFLAG ≠ 0 and bit 3 in CARDS = 1

<u>P Card 5</u>

All floating point

Col.	1-10	W + 12
	11-20	+ 13
	21-30	+ 14
	31-40	+ 15
	41-50	+ 16
	51-60	+ 17
	61-70	+ 18

The buffer W is used by the ADBASH subroutine. The values input will override the assembled values in these locations as the absolute error criteria.

SAVCON will be used to save the card for ELMOUT. If no errors, bit 4 in CARDS will be set = 1.

DESCRIPTION:
(continued)

P Card 6

Col. 1-9	JBUF	All floating point
10-18	JBUF+1	
19-27	JBUF+2	
28-36	JBUF+3	

If this card is input, it will override the assembled values in JBUF. Subroutine SAVCON will be used to save the card image for ELMOUT and bit 6 in CARDS will be set = 1 if no errors.

P Card 9

Col. 1-5	PRI+0	All floating point
6-8	1	
9-13	2	
14-18	3	
19-21	4	
22-26	5	
27-31	6	
32-34	7	
35-39	8	
40-44	9	
45-47	10	
48-52	11	
53-57	12	
58-60	13	
61-65	14	
66-70	15	
71-73	16	
74-78	17	

DESCRIPTION:
(continued)

This card contains the intervals desired to be printed or put on an empheris tape in a prediction.

One pass is made through the buffer to count the number of sets of intervals (3 fields = a set).

Then the buffer is checked to set that the times are in order. If not, an error exit is taken.

If the times are in order, PRT + 18 will contain the number of sets of intervals (T47) and bit 9 in CARDS will be set = 1.

P Card 10

Col.	1-15	GIT + 1,2	All BCD
	16-30	+ 3,4	
	31-45	5,6	
	46-60	7,8	
	61-75	9,10	

Each set of 2 words is checked for blanks. If not blanks, use FSKLOK to convert to floating point days and fraction. If all fields are blank, go to error exit.

Check to see that times are in order. If not, take error exit; otherwise, set bit 10 in CARDS = 1.

P Card 7

Col.	1-9	CSNM + 0	All floating poir
	10-18	+ 1	
	19-27	+ 2	
	28-36	+ 3	
	37-45	+ 4	
	46-54	+ 5	
	55-63	+ 6	
	64-72	+ 7	

DESCRIPTION:
(continued)

P Card 8

Col.	1-9	CSNM + 8
	10-18	+ 9
	19-27	+ 10
	28-36	+ 11
	37-45	+ 12
	46-54	+ 13
	55-63	+ 14
	64-72	+ 15

All floating point

These cards contain the tesseral coefficients for the MARTINI subroutine. SAVCON is used to save these card images for ELMOUT. If either or both cards are input, TESRAL is set = 1.

If no errors on P Card 7 & 8 bits 7 and 8 respectively will be set = 1 in CARDS.

Having unpacked and validated the input parameter cards, further checks are made:

If neither P3 nor P4 was input, ie, DCFLAG & PPPFLAG, = 0, the job cannot be run.

If neither or only one of P1 and P2 were input, then the elements will be retrieved from EBLOC and a comment printed. If EBLOC has no elements, the job will be terminated.

If PREDFLG = 0,

PRTIM will be made minutes since epoch by a double precision operation of:

PRTIM = (PRTIM-ORGDA) X 1440 + (PRTIM+1 - ORGTM)

If FTFLAG = 0,

PRTIME will be made minutes since epoch in the same manner as PRTIM.

DESCRIPTION:
(continued)

Using FYKLOK, ORGDAY will be computed as days from beginning of year to epoch.

If EPREV $\geq$ 100,000 it will be modulated 100,000 as a 5 digit revolution number is maximum.

If TESRAL $\neq$ 0, ie, P7 and or P8 were input, location CARDS will be tested to see that both were input. If not, a comment will be printed and the job terminated, as both or neither are required for the MARTINI subroutine.

Having successfully passed all preceding checks, location CARDSW will be tested. If = 0, no errors were found on any card, and the job may be run. (EXIT + 2H). If CARDSW $\neq$ 0, one or more cards were in error, and the job will be terminated (EXIT + 1H).

PURPOSE:	To initialize ephemeris tape for output.
CALL SEQUENCE:	JMP ITAPEW
INPUT:	SATNM SATNM + 1 } Satellite Name BCD
OUTPUT:	1 block on tape 10 See description. TAPCNT = C/HLT, TAPBUF
SUBROUTINES:	Program - TAPEW System - SYS, SYSNO
STORAGE REQUIREMENTS:	8 Cells
DESCRIPTION:	Positions tape to 1st block writes 1 block on tape (1st 2 words are the satellite name - last 2 words are zero), which fulfills requirements of the XYZLA program.

PURPOSE: To compute perturbative effects due to the atmosphere.

CALL SEQUENCE:

```
JMP   JNDRAG
JMP   (Error), h < LOLIMIT
(Normal Return)
```

INPUT:

DFLAG = 0 do not compute drag perturbations
= 1 compute drag perturbations

LOLIMIT = 50 km for D.C. and Prediction
= 10 km only for Cowell option

TEMPO = Exospheric temperature at epoch from TEMP subroutine

A_p - from P Card 3 or P Card 4

B = $C_D A/_m$

KAPPA - from P Card 3 or P Card 4

ASCON2 = F/6378.165

THDOT = $\dot{\theta}$

Output from SUBXYZ routine at time t.

OUTPUT:

XDTGR = $\ddot{x}_D$

YDTGR = $\ddot{y}_D$

ZDTGR = $\ddot{z}_D$

SUBROUTINES:

[illegible] - FASIN, FSIN, FSIN, FSQRT, FTENX
Program - CALH, ANGSUN

STORAGE REQUIREMENTS: 337 Cells

DESCRIPTION:

This subroutine uses the Nicolet II (1964) dynamic atmosphere tables from 120 km. to 1000 km., and the Coesa (1962) tables from 0 km. to 120 km. The table at location NICOLET is comprised of packed octal words for 10 km. altitude increments.

This subroutine is called by DERIV and CDERIV subroutines, however, there is a special entrance (JNDRAGI) and exit (JNSW) for subroutine CDLTB, which uses JNDRAG to compute log ρ (h_i, T), where i = 1, 2, 3, 4 in order to compute density scale height.

The subroutine performs the following functions:

(1) Correct for latitude and longitude from sun. The ANGSUN subroutine computes the position of the sun, α_o and δ_o.

$$\emptyset = \sin^{-1} U_z$$

$$\theta = \tan^{-1}\left(\frac{U_y}{U_x}\right)$$

(2) Compute log ρ (h_i, T) for four altitudes.

(3) Interpolate for log ρ (h, T) from results of (2).

(4) Compute ρ from log ρ (h,T)

(5) Compute the velocity relative to the atmosphere:

$$v_x = \dot{x} + \dot{\theta}y$$

$$v_y = \dot{y} - \dot{\theta}x$$

$$v_z = \dot{z}$$

$$v = \sqrt{v_x^2 + v_y^2 + v_z^2}$$

DESCRIPTION:
(continued)

(6) Compute the perturbative accelerations due to drag.

$$\text{XDTGR} = \dot{x}_D^{\backprime} = \nu_x \; (B \; \rho \; K_B \; \frac{\nu}{2} \; 6378.165)$$

$$\text{YDTGR} = \dot{y}_D^{\backprime} = \nu_y \; (B \; \rho \; K_B \; \frac{\nu}{2} \; 6378.165)$$

$$\text{ZDTGR} = \dot{z}_D^{\backprime} = \nu_z \; (B \; \rho \; K_B \; \frac{\nu}{2} \; 6378.165)$$

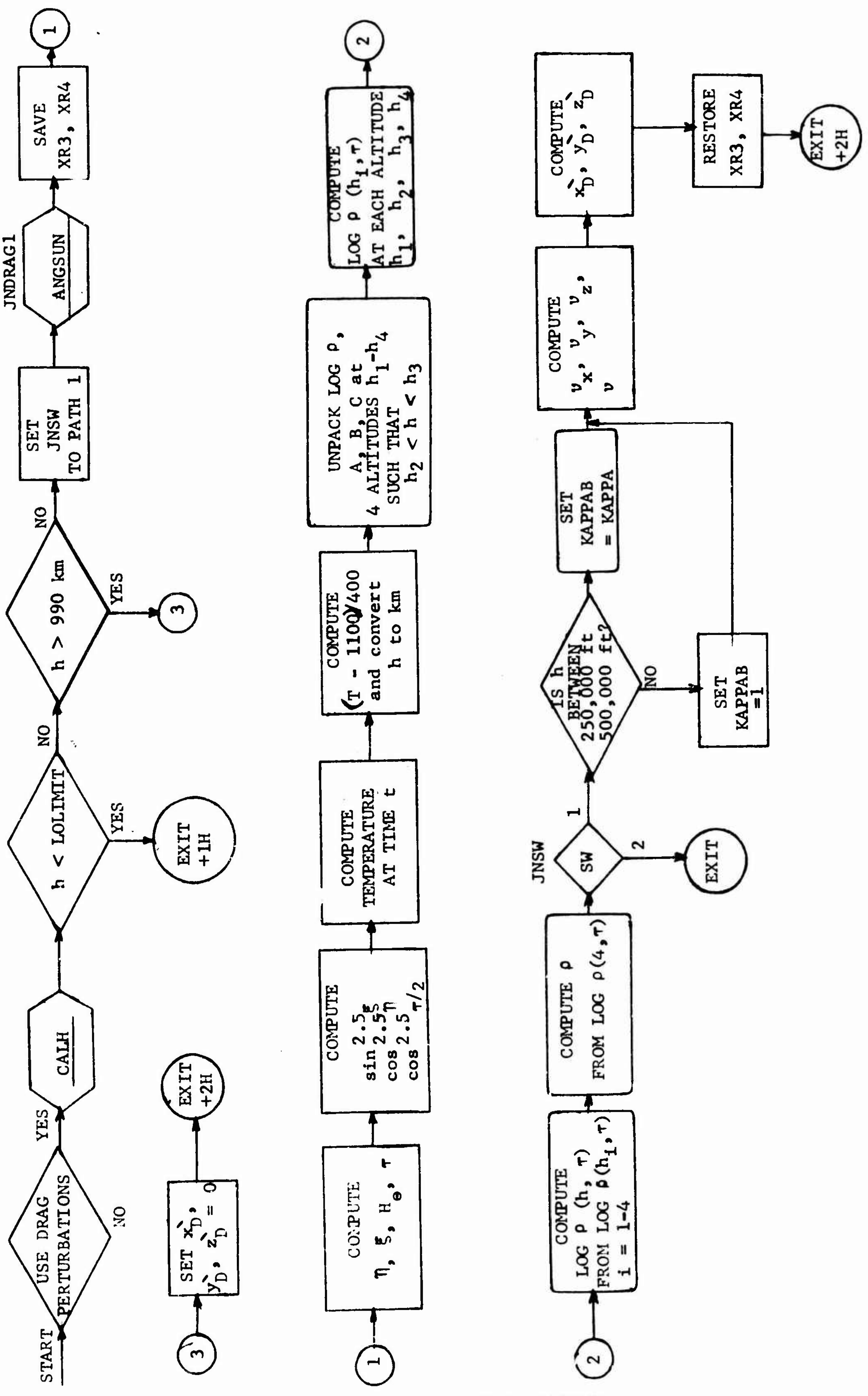


FIGURE 7. JNDRAG FLOW DIAGRAM

LPART
SPIRDEC

PURPOSE: Given a quantity, X in radians, this subroutine will compute Y and Z such that:

1. Y and Z have the same sign as X
2. Y is exactly divisible by 2π
3. $-2\pi \leq Z \leq 2\pi$
4. $Y + Z = X$

CALL SEQUENCE:

```
TMA    X
JMP    LPART
```

INPUT: The input consists only of the quantity X in the A register in floating point.

OUTPUT: Upon return from LPART, the A register contains Y, and the Q register contains Z. Both Y and Z are in floating point.

SUBROUTINES: None

STORAGE REQUIREMENTS: 14 Words

LSQ
SPIRDEC

PURPOSE:	To zero matrix buffer A11 to A11 + 48 and B11 to B11 + 6
CALL SEQUENCE:	JMP LSQ
INPUT:	A11 and B11 buffers must be located sequentially in core.
OUTPUT:	F/0 in buffer A11 to A11 + 48 and B11 to B11 + 6
SUBROUTINES:	(None)
STORAGE REQUIREMENTS:	4 Cells
DESCRIPTION:	See purpose.

PURPOSE: To form a least squares matrix.

CALL SEQUENCE: JMP LSQF

INPUT: C_1 thru C_{n+1} in cells TERMS thru TERMS + n

OUTPUT: Modified matrix in A11 buffer and B11 buffer.

SUBROUTINES: None

STORAGE REQUIREMENTS: 15 cells

DESCRIPTION: Add values in TERMS buffer to matrices A (A11 buffer) and B (B11 buffer).

$$A=\begin{bmatrix} \Sigma C_1\, C_1 & \Sigma C_1\, C_2 & \cdots & \Sigma C_1\, C_n \\ \Sigma C_2\, C_1 & \Sigma C_2\, C_2 & & \Sigma C_2\, C_n \\ \vdots & \vdots & & \vdots \\ \Sigma C_n\, C_1 & \Sigma C_n\, C_2 & \cdots & \Sigma C_n\, C_n \end{bmatrix} \qquad B=\begin{bmatrix} \Sigma C_1\, C_{n+1} \\ \Sigma C_2\, C_{n+1} \\ \vdots \\ \Sigma C_n\, C_{n+1} \end{bmatrix}$$

PURPOSE:	To solve the matrix equation AX = B for X.
CALL SEQUENCE:	JMP LSQS
INPUT:	A matrix in A11 buffer B matrix in B11 buffer
OUTPUT:	Solution X in LSQX buffer
SUBROUTINES:	Philco - FMAIN, FMAMU Program - MATRIX
STORAGE REQUIREMENTS:	15 cells
DESCRIPTION:	The top half of matrix A is accumulated at each entry to LSQR. When entry is made to LSQS, the terms of A are moved to the bottom half. Then the Philco matrix inversion subroutine (FMAIN) and matrix multiplication subroutine (FMAMU) are used to solve the equation for X. The solution is left in buffer LSQX through LSQX+(n-1). After the A matrix is inverted, subroutine MATRIX is called to solve for the standard deviations and correlation matrix of the delta elements. (See LSQR for matrix definition.)

PURPOSE: To compute the perturbative effects of the earth's bulge.

CALL SEQUENCE: JMP MARTINI

INPUT:

BFLAG $\begin{cases} = 0 \text{ do not compute perturbations} \\ = 1 \text{ use perturbations} \end{cases}$

W + 1 = t(min)

AE = a_e X = x

UZ = U_z Y = y

R = r THGR = θ_{gr}

TESRAL $\begin{cases} = 0\text{, use only zonals} \\ = 1\text{, use zonals \& tesserals} \end{cases}$

Buffers: JBUF, CSNM

OUTPUT:

XBDGR = $\dot{x}'_B$

YBDGR = $\dot{y}'_B$

ZBDGR = $\dot{z}'_B$

and values listed under description

SUBROUTINES:

System - ARCTAN
Philco - FSIN, FCOS, FSQRT

STORAGE REQUIREMENTS: 169 Cells

DESCRIPTION:

RTOMUZ2 = $\sqrt{1-U_z^{\,2}}$

If $\sqrt{1-U_z^{\,2}} \neq 0$	If $\sqrt{1-U_z^{\,2}} = 0$
SSUBX = $S_x = U_x U_z/\sqrt{1-U_z^{\,2}}$	S_x = F/0
SSUBY = $S_y = U_y U_z/\sqrt{1-U_z^{\,2}}$	$S_y = -U_z$
SSUBZ = $S_z = -\sqrt{1-U_z^{\,2}}$	S_z = F/0

DESCRIPTION:
(continued)

If $\sqrt{1-U_z^2} \neq 0$	If $\sqrt{1-U_z^2} = 0$
ESUBX = E_x = $-U_y/\sqrt{1-U_z^2}$	E_x = F/1
ESUBY = E_y = $U_x/\sqrt{1-U_z^2}$	E_y = F/0
ESUBZ = E_z = F/0	E_z = F/0
ZSUBX = Z_x = U_x	Z_x = U_x
ZSUBY = Z_y = U_y	Z_y = U_y
ZSUBZ = Z_z = U_z	Z_z = U_z

Compute P_n and P'_n for zonals for n = 2, 3, 4, 5

1) $P_n = 1/n \left[(2n-1) \; U_z P_{n-1} \; - \; (n-1) \; P_{n-2} \right\}$

where P_o = F/1, and $P_1 = U_z$

$P'_n = U_z \; P_{n-1} + n \, P_{n-1}$

where P'_1 = F/1

$QQ = a_e/r$

$QQ + 1 = \mu/r^2$

If TESRAL = 0, only compute zonal effects:

$$\left.\begin{array}{l} \text{2) GE} = g_{et} = \text{F/0} \\ \text{GS} = g_{st} = \text{F/0} \\ \text{GU} = g_{ut} = \text{F/0} \end{array}\right\} \quad \text{tesseral effects are set} = 0$$

DESCRIPTION: (continued)

3) $$GUZ = g_{uz} = \frac{\mu}{r^2} \sum_{n=2}^{5} (n+1) \; J_n \left(\frac{a_e}{r}\right)^n P_n$$

$$GEZ = g_{ez} = F/0$$

$$GSZ = g_{sz} = \frac{\mu \sqrt{1-U_z^2}}{r^2} \sum_{n=2}^{5} J_n \left(\frac{a_e}{r}\right)^n P'_n$$

where $J_2 - J_5$ are stored in JBUF + 0 → JBUF + 3 consecutively

4) $$XBDGR = \ddot{x}_B = (g_{uz} + g_{ut}) \; Z_x + (g_{ez} + g_{et}) \; E_x + (g_{sz}+g_{st}) \; S_x$$

$$YBDGR = \ddot{y}_B = (g_{uz} + g_{ut}) \; Z_y + (g_{ez} + g_{et}) \; E_y + (g_{sz}+g_{st}) \; S_y$$

$$ZBDGR = \ddot{z}_B = (g_{uz} + g_{ut}) \; Z_z + (g_{ez} + g_{et}) \; E_z + (g_{sz}+g_{st}) \; S_z$$

(and exit)

If TESERAL ≠ 0, use the tesseral harmonics. Substitute the following for step (2) and go to step (3).

Compute R_{nm} and R'_{nm} for all $2 \leq n \leq 4$ and $m \leq n$, where

$$R_{nm} = \frac{P_{nm}}{\sqrt{1-U_z^2}}$$

and $R'_{nm} = \sqrt{1-U_z^2} \; P'_{nm}$

and are solved as follows:

DESCRIPTION:
(continued)

$\underline{R_{22}, R_{33}, R_{44}}$: $R_{mm} = (2m - 1)\sqrt{1-U_z^2}\ R_{m-1,\ m-1}$

where $R_{11} = F/1$

$\underline{R_{21}, R_{31}, R_{41}}$: $R_{n,1} = P_n'$

$\underline{R_{32}, R_{43}}$: $R_{n,\ n-1} = (2n-1)\ U_z\ R_{n-1,\ n-1}$

$\underline{R_{42}}$: $R_{nm} = \frac{1}{n-m}\left[-(n+m-1)\ R_{n-2,m} + (2n-1)\ U_z\ R_{n-1,m}\right]$

This formula is the general expression for R_{nm} but is only used for R_{42} to save program space and time.

$\underline{R'_{nm}}$: $R'_{nm} = \sqrt{1-U_z^2}\ R_{n,m+1} - mU_z\ R_{nm}$

where $R_{n,n+1} = F/0$

The values of R_{nm} are stored in buffer RBUF+0 - RBUF+8, and for R'_{nm} in buffer RPBUF+0 - RPBUF+7.

TEMP2 = $\lambda E = \tan^{-1}(y/x) - t\ (\text{RPTIM}) - \theta_{gr}$

Then build buffer SINCOS containing cos λ, sin λ .... cos 4λ, sin 4λ consecutively in SINCOS+0 - SINCOS+7.

DESCRIPTION:
(continued)

Solve the following for the tesseral harmonics:
where $2 \leq n \leq 4$ and $m \leq n$

$$GU = g_{ut} = - \frac{u}{r^2} \sqrt{1-U_z^2} \sum_{n=2}^{4} \sum_{m=1}^{n} (n+1) \left(\frac{a_e}{r}\right)^n R_{nm} (C_{nm} \cos m\lambda + S_{nm} \sin m\lambda)$$

$$GE = g_{et} = - \frac{u}{r^2} \sum_{n=2}^{4} \sum_{m=1}^{n} \left(\frac{a_e}{r}\right)^n R_{nm} (C_{nm} \sin m\lambda - S_{nm} \cos m\lambda) (m)$$

$$GS = g_{st} = - \frac{u}{r^2} \sum_{n=2}^{4} \sum_{m=1}^{n} \left(\frac{a_e}{r}\right)^n R'_{nm} (C_{nm} \cos m\lambda + S_{nm} \sin m\lambda)$$

Where C_{22} S_{22} ..... C_{44} S_{44} are stored consecutively in CSNM+0-CSNM+15. These values must be input on P cards 7 and 8 and will be a number or F/O.

Having solved these equations, go to step (3).

MATRIX
SPIRDEC
1 of 2

PURPOSE:	To compute a correlation matrix and the standard deviations of the delta elements.
CALL SEQUENCE:	JMP MATRIX
INPUT:	A^{-1} matrix from LSQR subroutine in buffer A11 COUNTR = C/HLT, 0; C/HLT, N N = Matrix Size $1 \leq N \leq 7$
OUTPUT:	Standard deviations in buffer SIGN Correlation matrix in buffer MATRIXB (See description)
SUBROUTINES:	Philco - FSQRT
STORAGE REQUIREMENTS:	55 Cells
DESCRIPTION:	To avoid destroying the A^{-1} matrix in A11 buffer, the half matrix is moved to buffer MATRIXB.

The variance-covariance matrix is defined by

$$\frac{C^2_{ij}}{\sqrt{C_{ii}C_{jj}}} \quad \text{where } C_{ij} \text{ are elements of } A^{-1}.$$

To compute this, each column is divided by the root of the diagonal term, and then each row is divided by the root of the diagonal. The roots of the diagonal terms (which are the standard deviations) are stored in buffer BUF and are then moved to the appropriate locations in buffer SIGN for output purposes. The position in the buffer is dependent on the elements being corrected. (See PRNTMAT subroutine for buffer positions). MATRIX is called by subroutine LSQS.

DESCRIPTION:
(continued)

The correlation matrix is defined:

$$\begin{array}{ccc}
\dfrac{C^2_{11}}{C_{11}\,C_{11}} & & \\[2ex]
\dfrac{C^2_{12}}{C_{11}\,C_{22}} & \dfrac{C^2_{22}}{C_{22}\,C_{22}} & \\[2ex]
\vdots & \vdots & \ddots \\[2ex]
\dfrac{C^2_{1j}}{C_{11}\,C_{jj}} & \dfrac{C^2_{2j}}{C_{22}\,C_{jj}} & \dfrac{C^2_{jj}}{C_{jj}\,C_{jj}}
\end{array} \qquad 1 \leq j \leq 7$$

The values are stored by column in MATRIXB.

MOD2PI
SPIRDEC

PURPOSE:	To modulate a number between 0 and 2 π radians.
CALL SEQUENCE:	TMA (number) JMP MOD2PI TAM (number)
INPUT:	(A) reg = number to be modulated 2 π TWOPI = 2 π radians
OUTPUT:	(A) = number (mod 2 π)
SUBROUTINES:	(None)
STORAGE REQUIREMENTS:	3 Cells
DESCRIPTION:	Adds or subtracts 2 π radians from the number in the (A) reg until it is between 0 and 2 π radians.

MOVBUF
SPIRDEC
1 of 2

PURPOSE: To move the observation buffer.

CALL SEQUENCE:

```
TMA  OCOUNT
JMP  MOVBUF
```

INPUT: Observations in 10 word format and in 6 word format starting in location EBLOC.

Left address of OBSREJ = number of observations rejected

Left address of OBSLEFT = number of observations in 10 word format

Left address of OBSPROC = total number of observations processed

Left address of OCOUNT = number of observations processed to 6 word format in one group

OAMDE = C/HLT, OBLOC; C/HLT, EBLOC

OUTPUT: See description

STORAGE REQUIREMENTS: 13 Cells

DESCRIPTION: Because of core limitations, the observations in the 10 word format of OBLOC are converted to a 6 word format (see below). Initially up to 984 observations (10 words/observation) are in EBLOC.
Subroutine PROOBS controls the formatting of the observations into 6 word entries. Subroutine MOVBUF is called by PROOBS and OBVEC (which is called by PROOBS).

Procedure:

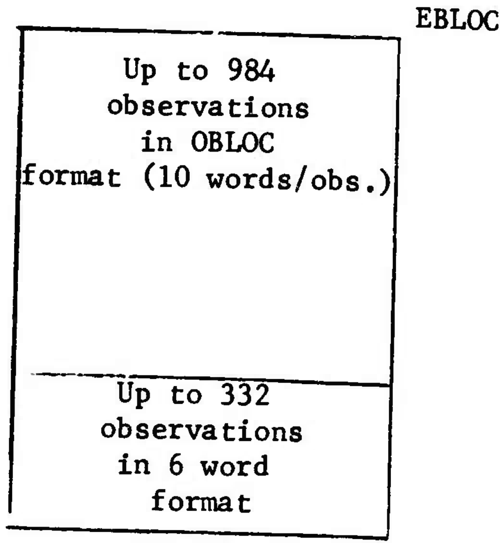
(1) Add the number of observations that have been converted to a 6 word format (OCOUNT) to the number of observations rejected (OBSREJ). Multiply this number by 10 and set up the move instructions at location MOVSW, if the result is less than 4096; otherwise print "more than 77 observations rejected" and exit.

(2) Multiply the number of observations still in the 10 word format (OBSLEFT) by 10 and add it to the total number of observations in a 6 word format (OBSPROC) which is multiplied by 6. This number is stored in OCOUNT as the total number of cells to be moved up.

DESCRIPTION:
(continued)

(3) After moving the buffer up,
set OCOUNT = 0
OBSREJ = 0
WOBMARK = C/HLT, OBLOC; C/HLT, EBLOC

(4) Store Z's in the next available location at the end of the buffer and move this location to Index Register 0.



PURPOSE: To retrieve next processed observation.

CALL SEQUENCE:
JMP MOVDAT
JMP (End of observations)
(Return)

INPUT:
Processed observations in EBLOC.
Current address in EBLOC from location SAVOBS.
Sensor data in SBUF.
Weights in WBUF.

OUTPUT:
STAID
OBFLG
T
CAPX
CAPY
CAPZ
CXDOT
CYDOT

$$\left.\begin{array}{l}\text{RANGE}\\\text{SIGMA1}\end{array}\right\}\quad\text{if range observed}$$

$$\left.\begin{array}{l}\text{ASUBX}\\\text{ASUBY}\\\text{ASUBZ}\\\text{DSUBX}\\\text{DSUBY}\\\text{DSUBZ}\\\text{XLSUBX}\\\text{XLSUBY}\\\text{XLSUBZ}\\\text{SIGMA3}\\\text{SIGMA4}\end{array}\right\}\quad\text{if angles observed}$$

$$\left.\begin{array}{l}\text{SIGMA2}\\\text{RODOT}\end{array}\right\}\quad\text{if range-rate observed}$$

SUBROUTINES: Program - GETSEN, SENLOC, AZREC, ALREC, GETWGT

STORAGE
REQUIREMENTS: 27 Cells

DESCRIPTION:

1) Unpacts the following from processed observation:

   STAID
   OBFLG
   T
   RANGE
   RODOT
   ALPHA
   DELTA

2) Use subroutine GETSEN to retrieve sensor data.

3) Use subroutine SENLOC to compute:

   THTA = θ
   SINTH = $\sin \theta$
   COSTH = $\cos \theta$

4) If azimuth and elevation were observed, call subroutine AZREC to compute $\underline{A}$, $\underline{D}$, $\underline{L}$.

5) If right ascension and declination were observed, use subroutine ALREC to compute $\underline{A}$, $\underline{D}$, $\underline{L}$.

6) Compute the following:

   CAPX = $X = (\cos \theta)\ (X/\cos \theta)$
   CAPY = $Y = (\sin \theta)\ (X/\cos \theta)$
   CXDOT = $\dot{X} = -Y \dot{\theta}$
   CYDOT = $\dot{Y} = X \dot{\theta}$

7) Retrieve the weights for this observation using subroutine GETWGT.

8) Update SAVOBS to next observation and exit.

PURPOSE: To retrieve observations and corresponding weights.

CALL SEQUENCE:

```
JMP   NXTOB
JMP   (End of Obs)
JMP   (No Sensor Data)
JMP   (No Weights)
JMP   (Normal Return)
```

INPUT:

OSTROB - right address = number of obs + 1
Observations in EBLOC
Weights in WBUF

OUTPUT:

STAID
T
DELTA
ALPHA
RANGE
RODOT
and output of subroutine GETSEN

SUBROUTINES: Program - GETSEN

STORAGE REQUIREMENTS: 39 Cells

DESCRIPTION:

(1) Unpact the following from the current observation in EBLOC:

STAID - 00000SSS
T
DELTA
ALPHA
RANGE
RODOT
} floating point
OTYPE = 0000000

(2) Search WBUF to find weights for the sensor number in STAID. If no match, exit +3H (No weights return). Otherwise go to step (3).

DESCRIPTION: (continued)

(3) Check to see that a weight is in WBUF for each observed quantity of the observation. If not, exit +3H.

(4) If all weights are entered, call subroutine GETSEN to retrieve the sensor data. If no sensor data, exit +2H. If sensor data, exit +4H (normal return).

Each call to NXTOB will retrieve the next observation, until an end of observation return is taken (+1H).

PURPOSE: To compute and/or store values in POBLOC for one observation at a time.

CALL SEQUENCE: JMP OBVEC

INPUT:

OBFLG = 0———0 XXXX (See PROOBS for description)

STAID = 00000 SSS Station number.

T = time (minutes since epoch)

$\left.\begin{array}{l}\text{RANGE}\\ \text{RODOT}\\ \text{ALPHA}\\ \text{DELTA}\end{array}\right\}$ floating point

OUTPUT:

See POBLOC format in PROOBS

T

THTA = θ_c

SINTH = sin θ

COSTH = cos θ

CAPX = X

CAPY = Y

and output of OBVECP, OBVECQ, AZREC, ALREC, RRATE

SUBROUTINES: Program - MOVBUF

STORAGE REQUIREMENTS: 15 Cells

DESCRIPTION:

1) An entry is stored in the new observation format in the address specified by index register 0:

Word 0	TOOOOSSS
1	time
2	range
3	range rate
4	alpha
5	delta

2) Add 1 to OBSPROC
Subtract 1 from OBSLEFT
Add 1 to MCOUNT

   a) if MCOUNT < 332, then exit

   b) if MCOUNT = 332, call subroutine MOVBUF to move the buffer up. (See MOVBUF for description.) Then exit.

PURPOSE: To keep a count (T15) of the lines output/page and force a new page with headings when count $\geq$ 55.

CALL SEQUENCE:

```
TMA   C/HLT,N
JMP   PAGECON
JAZ   (          )
```

N = No. of lines of output, set N = 55 to force page

(A) = 0 means new page was output

INPUT: See Call Sequence

OUTPUT: (1) Updated line count
and/or
(2) New page with headings when line count $\geq$ 55.

SUBROUTINES: Program-HEAD

STORAGE REQUIREMENTS: 7 Cells

DESCRIPTION: Updates line count (LINECNT) and jumps to subroutine HEAD (to output headings) when LINECNT $\geq$ 55. If the previous JMP to PAGECON put out a new page a second one cannot be forced.

PURPOSE: To control the prediction option.

CALL SEQUENCE:

```
JMP  PCONTRL
JMP  (Error)
(Normal)
```

INPUT: P Card 4
An element set from P Cards 1 and 2 or 6 Card element set or from the differential correction.

OUTPUT: Prediction ephemeris as requested - printed and/or written on a binary tape.
(See description)

SUBROUTINES:
System - FYKLOK
Program - ITAPEW, SUBOUT1, SEPSUB, INITIAL, SAV5PTS, TEMP, SAVW, SETW, SUBXYZ, RESW, TAPEW, FTAPEW, SUBOUT, ADBASH, REVSOB, DIVDIF, CALU, GIPAR

STORAGE REQUIREMENTS: 298 Cells

DESCRIPTION: This subroutine contains the logic necessary to control a prediction. There are several options; some can be combined but not all:

(1) Predict by revolution number or time.

(2) Output hard copy and/or binary tape.

(3) Prediction within one or more time intervals at specified time increments.

(4) Predict backward or forward in time from epoch.

(5) Request points (maximum of 5) to be left in core for a GIPAR run.

DESCRIPTION:
(continued)

Because several of these options can be combined, the logic is rather complicated.

A. The subroutine is initialized as follows:

- (1) Set several switches.
- (2) Initialize the buffer to be used by GIPAR (starts in EBLOC + 20).
- (3) Set switches controlling a prediction by time or revolution number.
- (4) Initialize the binary tape if requested.
- (5) Initialize the printed output if requested.
- (6) Move some values from P Card 4 to locations used. If the prediction follows a D.C., same results, such as B, override the input on P Card 4.
- (7) Test mode:
    - (a) If prediction only:
        - (1) Move more values from P Card 4
        - (2) Compute $\gamma B/2.2$
        - (3) Set REV = epoch revolution number
    - (b) If D.C. and prediction:
        - (1) Move new epoch elements from buffer to locations used.
        - (2) If time prediction, change final time from minutes since epoch to minutes since new epoch.
        - (3) Convert new epoch time from minutes since epoch to a base epoch.

DESCRIPTION:
(continued)

(4) Initialize page heading routine (IHEAD1) to print new epoch in page headings.

(5) Call subroutines THGRC, BEGIN.

(6) Set REV = new epoch revolution number.

(8) Initialize interpolation buffer:

(a) Call subroutine INITIAL to initialize ADBASH.

(b) Save initial elements as first set in the interpolation buffer.

(c) Set T = 0 (printed output time)
BT = 0 (binary tape output time)

(d) Call subroutine TEMP to compute temperature at epoch.

(9) Test if GIPAR points requested:

(a) If requested, prediction interval points will be ignored - if not, go to step A(10).

(b) Convert GIPAR time from days and fractions to minutes since epoch.

(c) Delete GIPAR points not between epoch (or new epoch) and the final time.

(10) Test if predicting forward or backward in time and set switches and locations accordingly.

(a) If predicting backward and if requesting prediction intervals, changes signs of interval buffer to negative.

DESCRIPTION (continued)

(11) Test if printed output requested:

- (a) If not, go to step B.
- (b) If yes, test if GIPAR points requested
  - (1) If not, go to A(11)(c).
  - (2) If requested, set a switch for GIPAR logic and move the first time requested to T.
- (c) Test if prediction intervals requested
  - (1) If not, go to step B.
  - (2) Otherwise, move start time, end time, and time increment to T, LTBUF+1 and ENDT respectively; also move start time to BT.

B. Basic integration loop:

(1) Call ADBASH to integrate to next point.

(2) Test if drag perturbations are greater than an epsilon; if so, switch to Cowell option for decay.

(3) Update revolution number if necessary.

(4) Save the new point in the interpolation buffer.

(5) If 5 points are in the buffer for interpolation, go to step B(6); if not, go to step B(1).

(6) Test if current revolution number equals final revolution number if a revolution prediction was requested:

- (a) if not equal, go to step C
- (b) if equal, go to step D(5).

DESCRIPTION (continued)

C Binary tape loop:

(1) If no tape output was requested go to step D.

(2) Test if binary tape time is within the time span of the interpolation buffer.

(a) If not, test if printed output was requested:

(1) Go to step B(1), if not requested.

(2) Go to step D, if requested.

(b) If in the range of the interpolation buffer

(1) Interpolate for elements at the time

(2) Use SUBXYZ subroutine to convert to $\underline{r}$, $\underline{r}$

(3) Call subroutine TAPEW to output the point

(4) Update the time

(5) Test if time or revolution prediction

(a) If time, test if 1st time ≥ final time; if not ≥ go to step C(2)(b)(5)(b); if equal, wrap up the binary tape and turn off tape option; if >, set t equal final time, and loop to C(2) once more.

(b) For revolution or time, test for prediction intervals; if not, go to step C(2); if yes, test if time > end time; if <, go to step C(2); if >, set time = end time and go to step C(2).

DESCRIPTION:
(continued)

D. Printed output loop:

- (1) If time prediction, test if time ≥ final time:
  - (a) If ≥, go to D(5).
  - (b) If <, go to step D(2).
- (2) Test if time is in span of interpolation buffer
  - (a) If not, go to step B.
  - (b) If yes, interpolate for elements at the time; convert to $\underline{r}$, $\dot{\underline{r}}$, and output the point.
- (3) If GIPAR points were requested, call subroutine GIPAR to convert and store the values
  - (a) If revolution prediction, go to D(3)(b)(2)(a).
  - (b) If time prediction - test if more times
    - (1) If not, turn off prediction interval option and go to D(5)(b)(for revolution prediction) or to D(5)(a) (for time prediction).
    - (2) If yes, compare next time to final time:
      - (a) if ≤ put next time in T and go to D(1)
      - (b) if > go to D(3)(b)(1).
- (4) If GIPAR points not requested, update time, and test for prediction intervals:
  - (a) If no intervals, go to D(2).
  - (b) If intervals:
    - (1) If time ≤ end time, go to D(2)

DESCRIPTION:
(continued)

(2) If time > end time, set time = end time and go to D(2) the first time; second time, check for end of intervals; if end, go to D(5)(b) (for revolution prediction), to D(5)(a) (for time prediction), if not end, move next interval times from buffer, reset switch and go to D(2).

(5) Switch at D(1)(a)

(a) First time, set time to final time, set switch to second time, go to step C.

(b) Second time, test if tape output:

wrap up tape, if yes
then exit +2H (normal exit)

E. Cowell option.

When a satellite decays to about 90 km. altitude, the drag coefficient is such that integration should continue in the Cowell mode.

(1) A test is made for backward integration, which is not allowed for decay.

(2) Since the interpolation buffer now contains $\underline{N}$,$\underline{M}$ elements, it is necessary to integrate backwards for sufficients points to replace the $\underline{N}$,$\underline{M}$ elements with $\underline{r}$, $\underline{\dot{r}}$, for the same time range.

(3) Set the integration to go forward integrating for $\underline{r}$ $\underline{\dot{r}}$ elements until the interpolation buffer contains 5 points.

(4) The first time the buffer is full, save the interpolation buffer, time for restart in the decay corridor.

(5) Test for tape output. If requested, test if the output time is in the range of the interpolation buffer. If so, interpolate for the elements, output the point, update the time. Continue looping until the time range is excluded, then go to E(6).

DESCRIPTION:
(continued)

(6) Test for printed output. Follow the same procedure as step E(5). Go to step E(3) when the time range is excluded.

The loop of steps(3)through(6)will continue until the CDERIV subroutine (called by ADBASH) exits to step E(7). This is done when the vehicle drops below 10 km. At this point, the decay corridor is produced.

(7) Print comment that vehicle decayed.

  (a) If first entrance, double the value of B and print a comment to this effect. If tape output was requested, wrap up the ephemeris tape and turn off the tape output option. Restore the values saved at step E(4) and go to step E(5) to produce an empheris for a drag coefficient of twice the original value.

  (b) If second entrance, multiply the original B by 1/2, print a comment and follow the same procedure as E(7)(a) only with a different value of B.

  (c) If third entrance, exit (+2H) the decay corridor and prediction are completed.

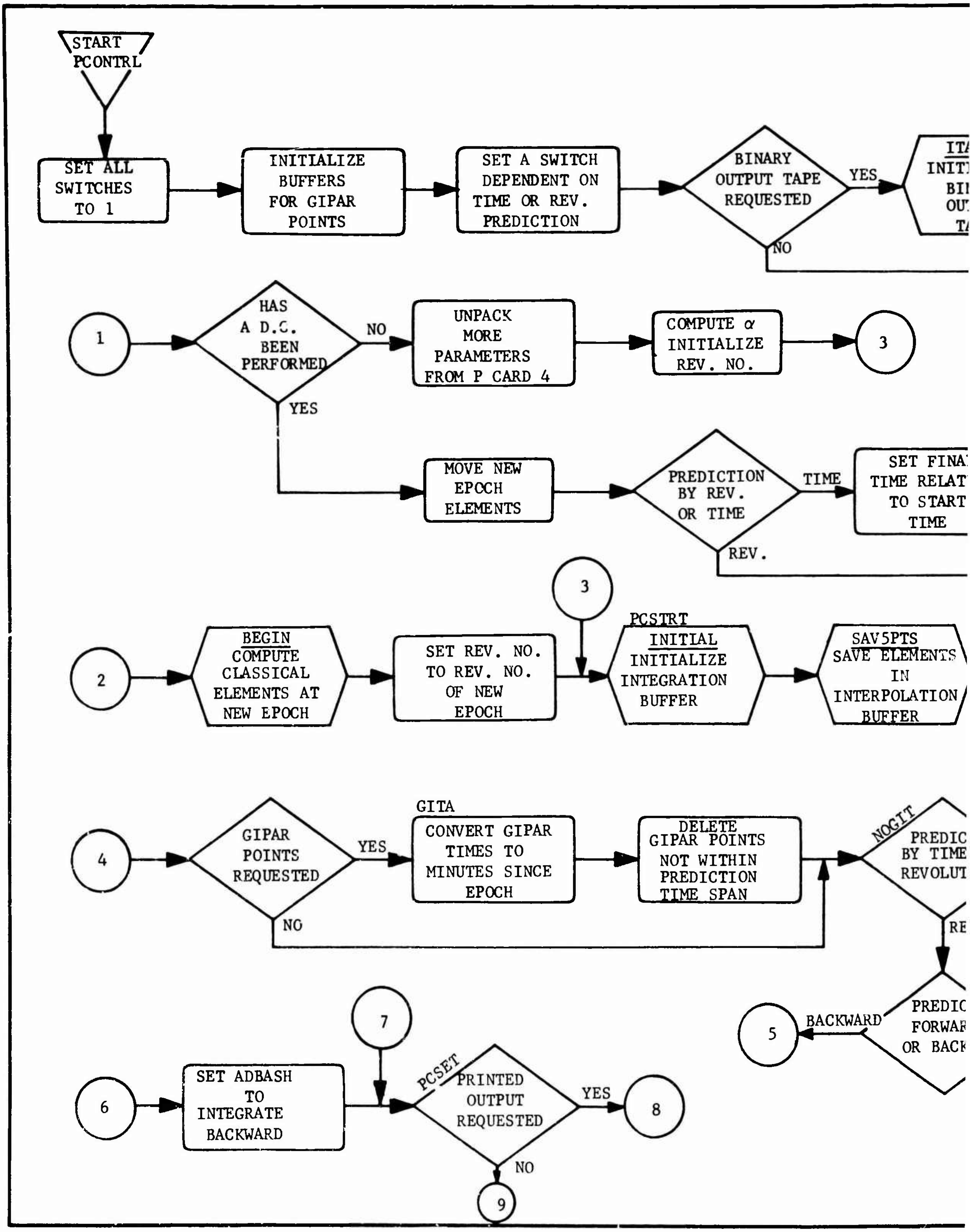


FIGURE 8. PCONTROL FLOW DIAGRAM (1 of 4

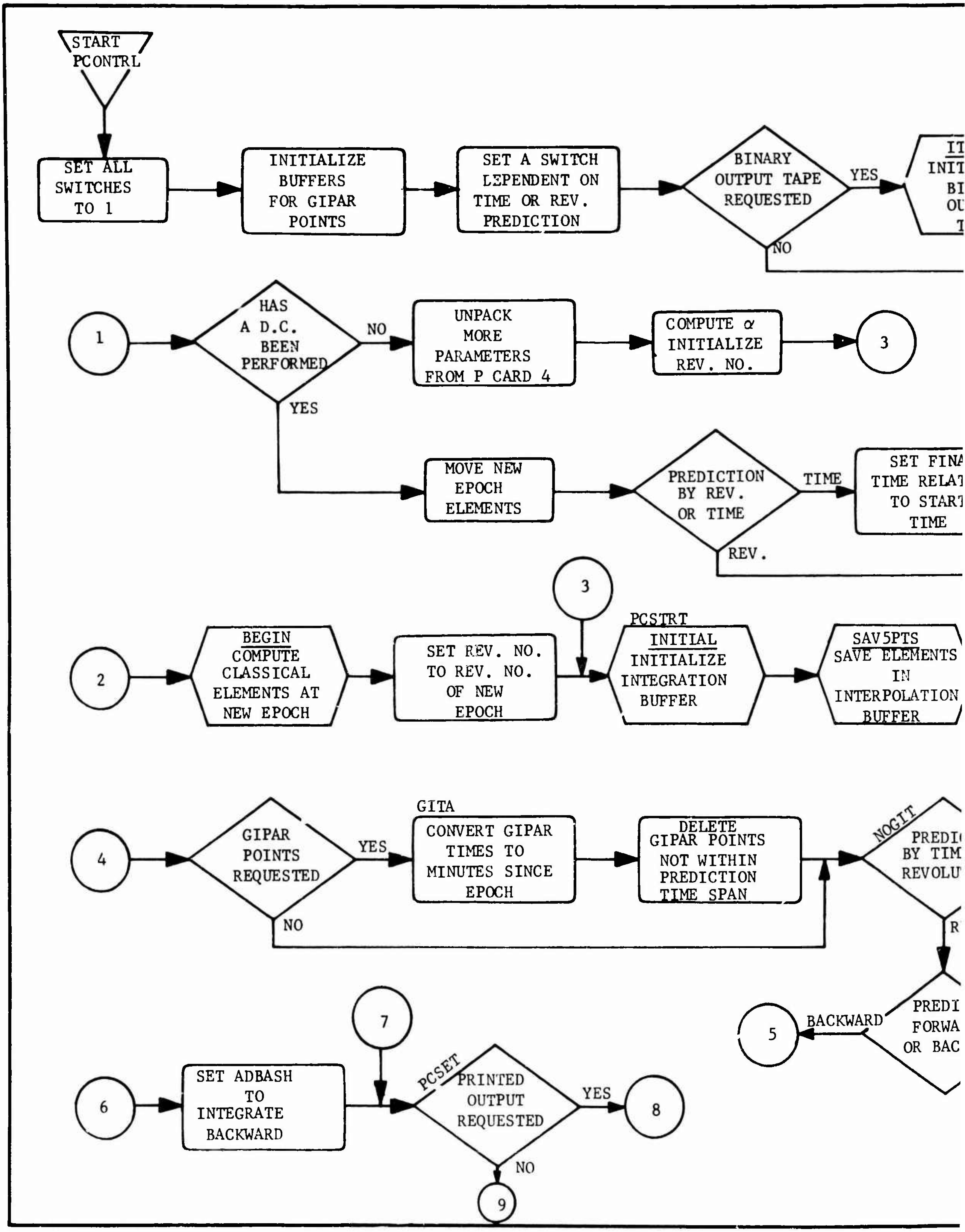
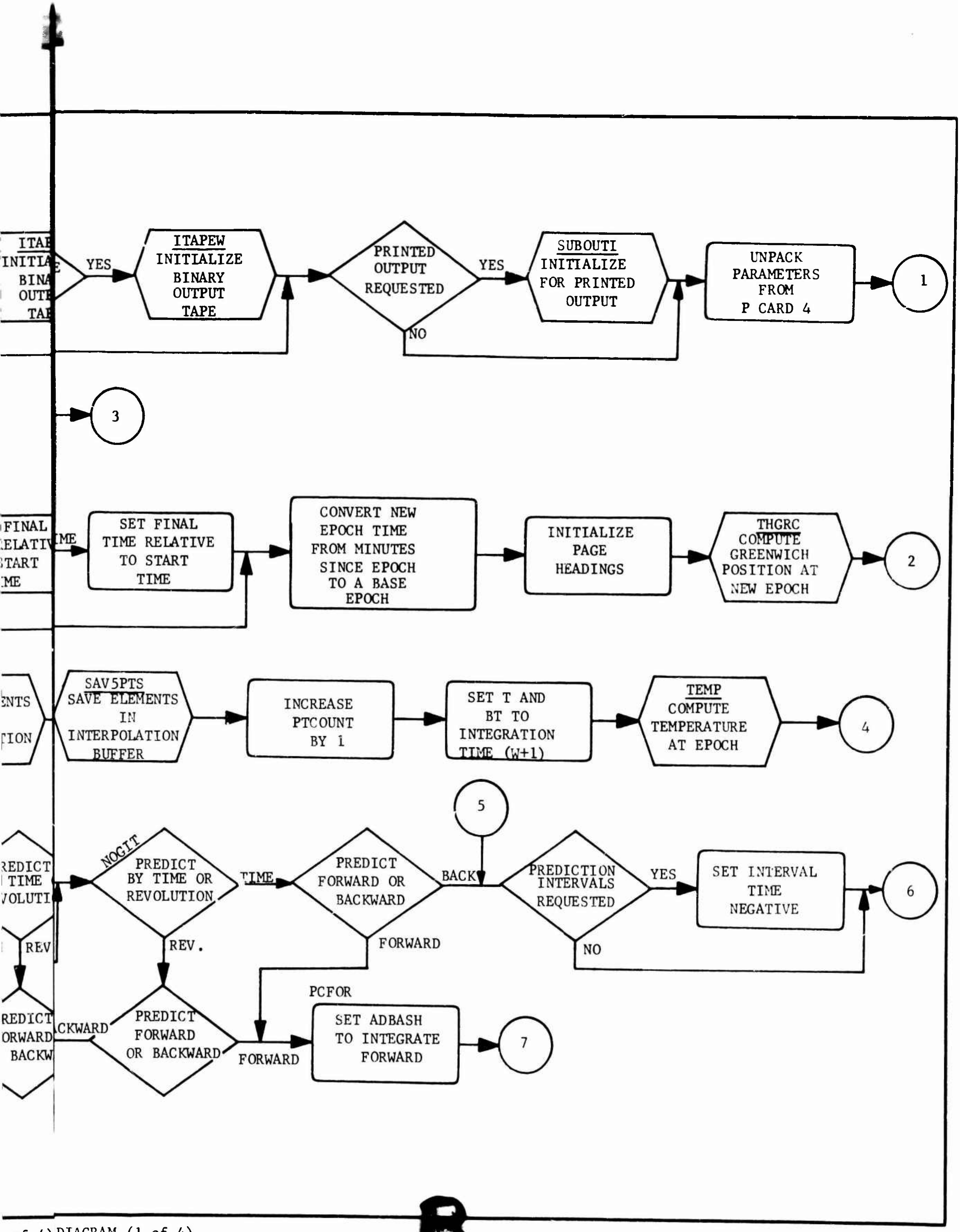


FIGURE 8. PCONTROL FLOW DIAGRAM (1 of



of 4) DIAGRAM (1 of 4)

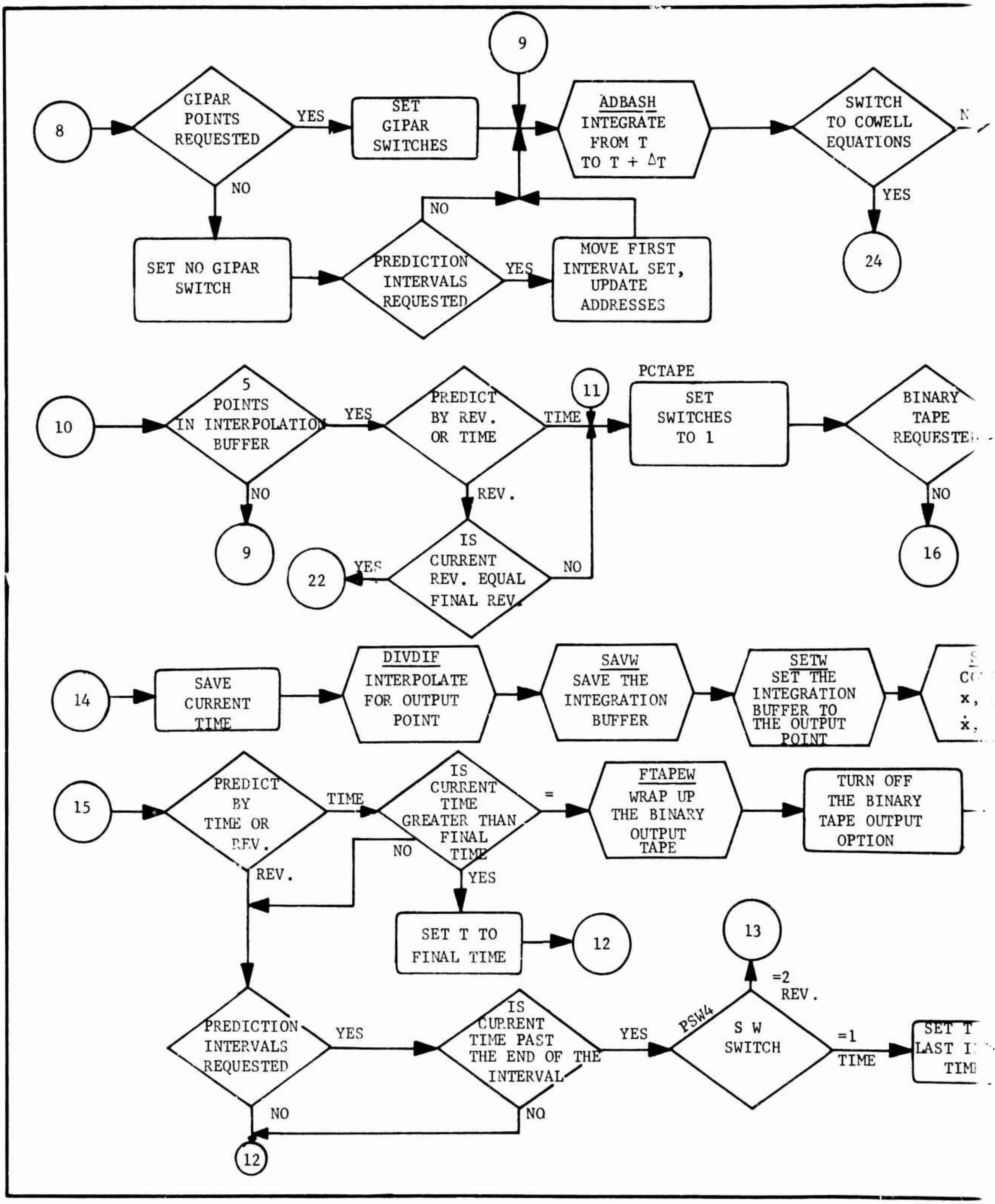


FIGURE 8. PCONTRL FLOW DIAGRAM (2 of 4)

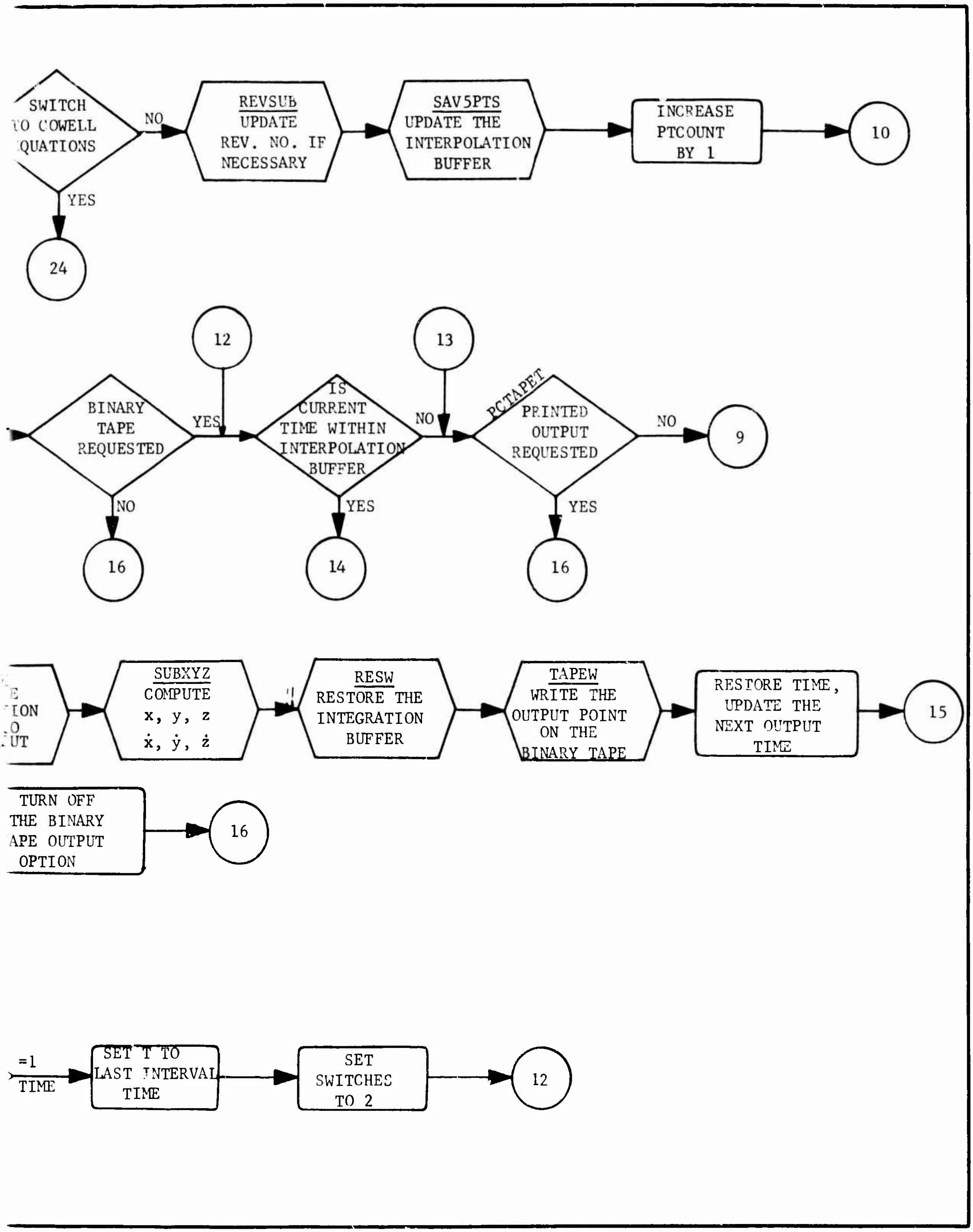


DIAGRAM (2 of 4)

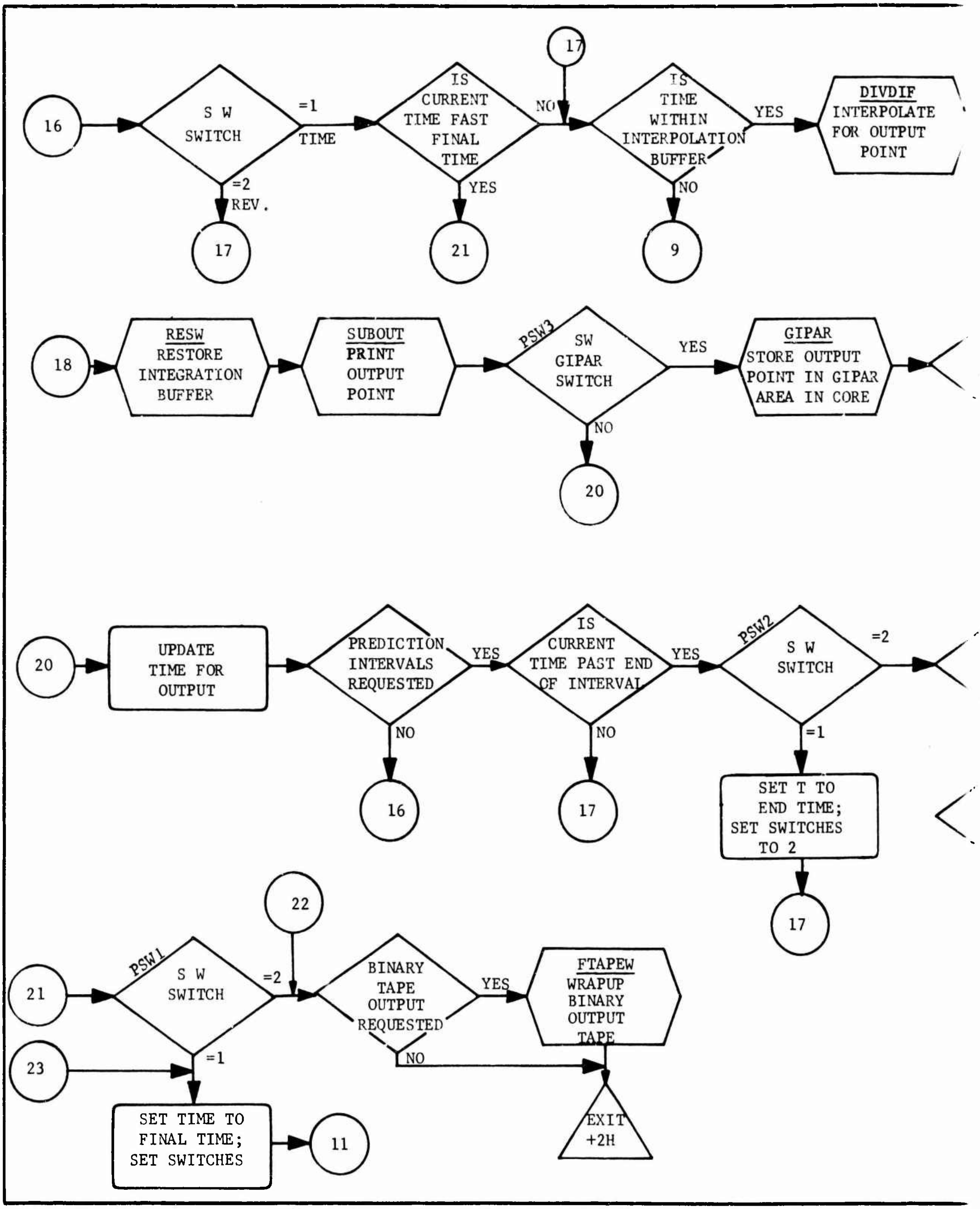
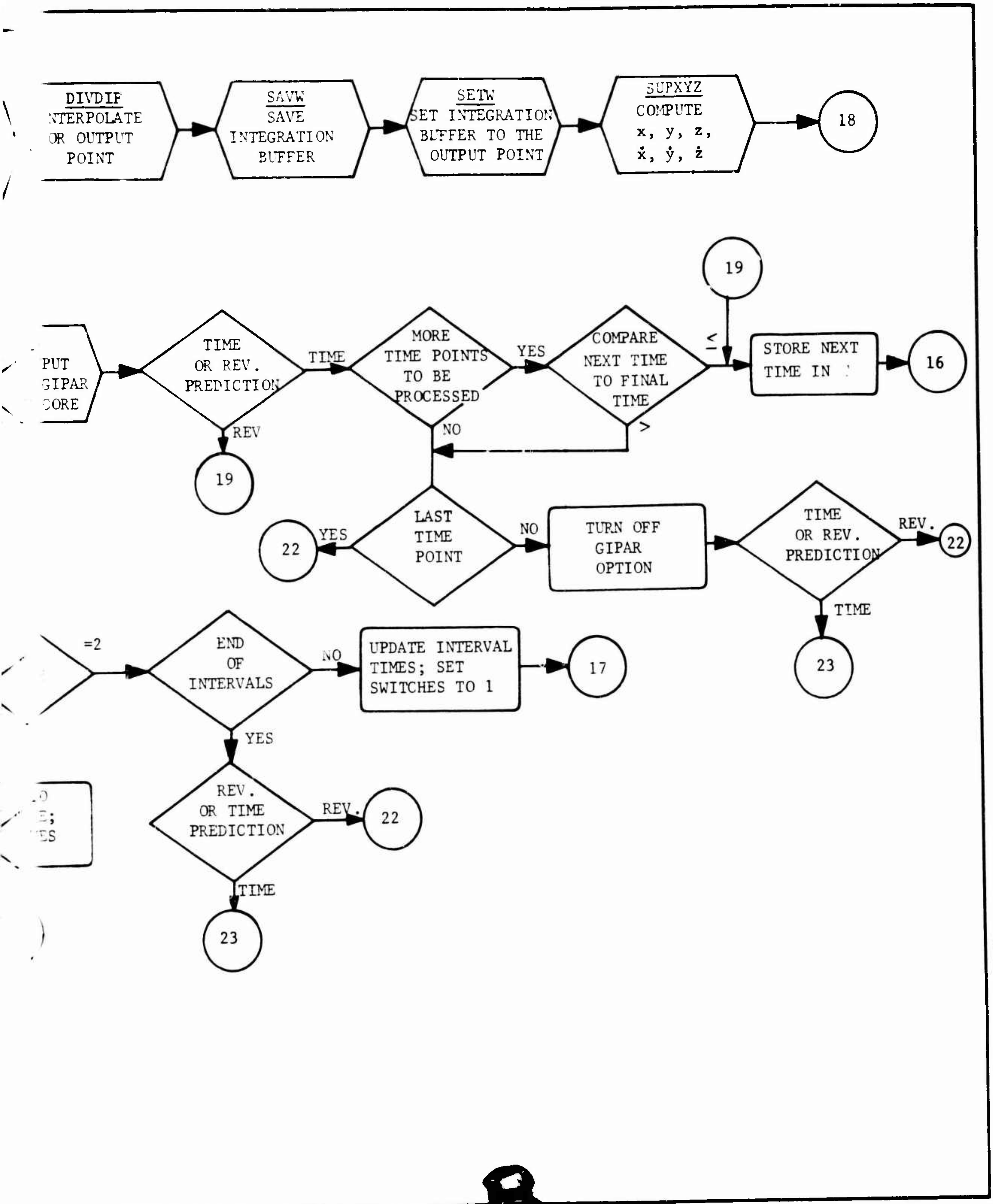
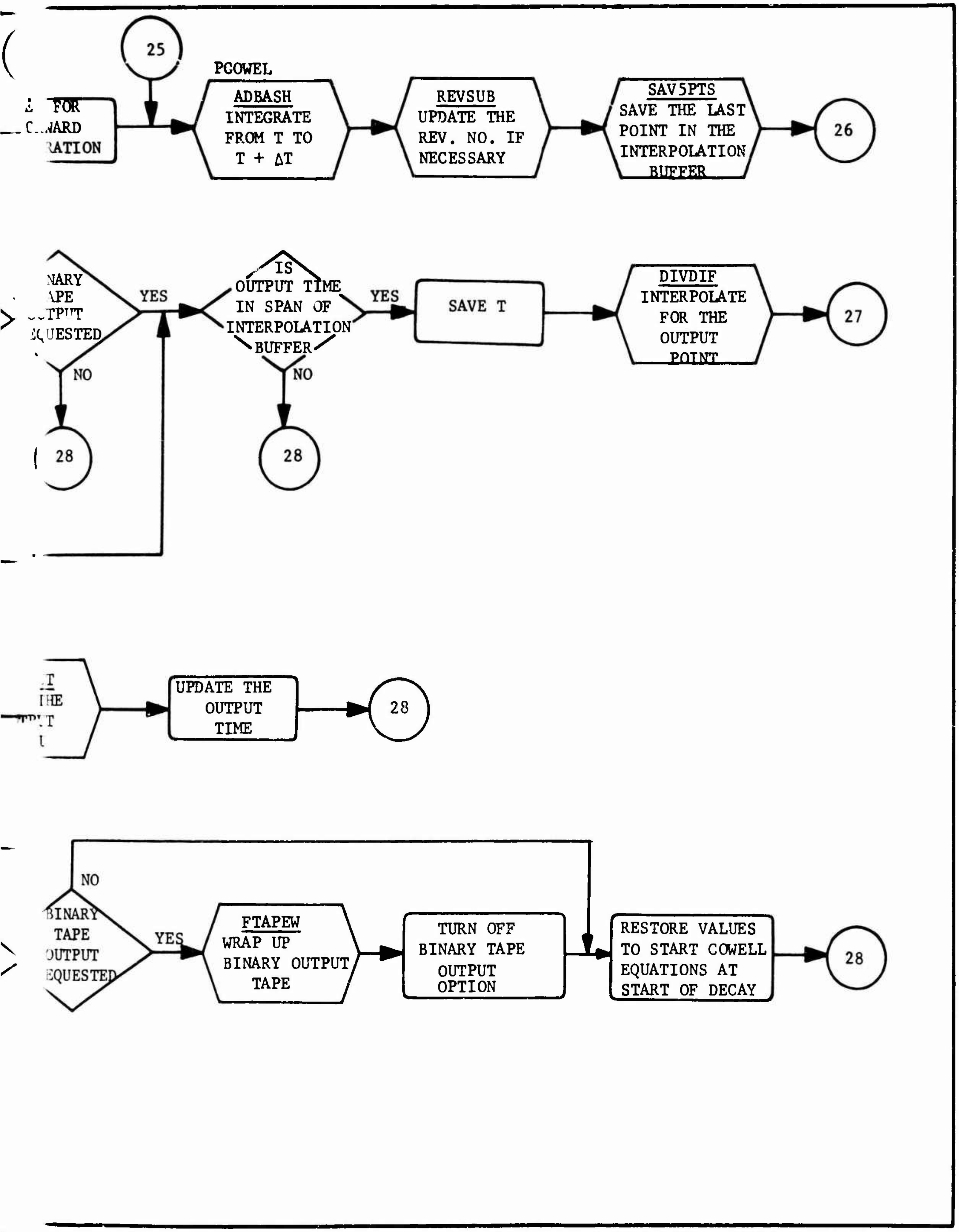


FIGURE 8. PCONTRL FLOW DIAGRAM (3 of 180



B

DIAGRAM (3 of 4)



FOR
C..WARD
RATION
25
PCOWEL
ADBASH
INTEGRATE FROM T TO T + ΔT
REVSUB
UPDATE THE REV. NO. IF NECESSARY
SAV5PTS
SAVE THE LAST POINT IN THE INTERPOLATION BUFFER
26
NARY
APE
UTPUT
EQUESTED
YES
NO
28
IS OUTPUT TIME IN SPAN OF INTERPOLATION BUFFER
YES
NO
28
SAVE T
DIVDIF
INTERPOLATE FOR THE OUTPUT POINT
27
T
THE
UPDATE THE OUTPUT TIME
28
NO
BINARY TAPE OUTPUT EQUESTED
YES
FTAPEW
WRAP UP BINARY OUTPUT TAPE
TURN OFF BINARY TAPE OUTPUT OPTION
RESTORE VALUES TO START COWELL EQUATIONS AT START OF DECAY
28

M (4 of 4)

PHLAH
SPIRDEC

PURPOSE: To compute φ, λ, h

CALL SEQUENCE: JMP PHLAH

INPUT:

UX = U_z
PI = π radians
XIMFSQ = $(1-f)^2$
TWOPI = 2 π radians
X = x
Y = y
RPTIM = rad/solar min.
W + 1 = t (min. since epoch)
THGR = θ_{gr}

OUTPUT:

PHI = φ (deg)
XLAMD = λ (deg) -180° (W) $< \lambda < +180^\circ$ (E)
H = h (km)
LFLAG = E or W for λ (BCD)

SUBROUTINE:

Program - MOD2PI, CALH
System - ARCTAN
Philco - FSQRT

STORAGE REQUIREMENTS: 18 Cells

DESCRIPTION:

$$PHI = \varphi = \tan^{-1}\left[U_z / \sqrt{1 - U_z^{\,2}} \quad (1 - f)^2\right]$$

$$THETA = \tan^{-1}(y/_x)$$

$$XLAMB = - t\ (.0043752691) - THGR + THETA$$

Uses CALH subroutine to compute h.

PURPOSE: To retrieve or compute the new epoch elements.

CALL SEQUENCE: JMP PRERES

INPUT:

SOLDUZ = last U_z from DC

SREV = last rev. no. from DC

PRDSAV = 0 means new epoch elements found in DC

= 1 continue integration until new epoch time

EPREV = epoch rev. no.

SAVELEM = 0; continue DC integration to compute elements

= 1; initialize integration to compute elements

and input required for subroutines.

OUTPUT:

LPRINT = L (deg)

AXPRINT = a_{xn}

AYPRINT = a_{yn}

HXPRINT = h_x

HYPRINT = h_y

HZPRINT = h_z

BPRINT = B

HSUBQP = h_q (km)

PAPRINT = p_a (min)

and time in BCD format.

SUBROUTINES: Program - INITIAL, RESWBF, RESICK, ADBASH, REVSUB, SAV5PTS, DIVDIF, SAVICK, SAVWBF, BCDTIM

STORAGE REQUIREMENTS: 51 Cells

DESCRIPTION: There are 2 main paths to take in this subroutine: (1) to retrieve the new epoch elements which were computed in the DC, i.e., the new epoch elements were in the span of the observations, (2) to continue the integration started by the DC until the new epoch time is reached.

DESCRIPTION:
(continued)

I. If the new epoch elements were found in the D.C. (PRDSAV = 0), the values that were saved at the time are restored and the rest of the values computed:

Restore: B00 + 0 = a_{xno} ⟶ AXPRINT

+ 1 = a_{yno} ⟶ AYPRINT

+ 2 = B ⟶ BPRINT

+ 9 = L (rad) ⟶ LPRINT (deg)

+13 = h_x ⟶ HXPRINT

+14 = h_y ⟶ HYPRINT

+15 = h_z ⟶ HZPRINT

Compute:

$$\text{HSUBQP} = h_q \text{ (km)} = \left\{a\ (1-e) - 1\right\}\ \frac{6378.165 \text{ km}}{\text{e.r.}}$$

where B00 + 3 = a

B00 + 4 = e^2

$$\text{PAPRINT} = P_a \text{ (min)} = 2\pi\left\{\frac{\frac{(1-.5e^2)}{p^2}\left(\frac{3}{2}\sin^2 i-1\right)\text{P3JA02} + 1}{n_o}\right\}$$

where B00 + 5 = sin i

B00 + 6 = p

B00 + 7 = n_o

Use subroutine BCDTIM to get time in BCD format, where B00 + 8 = t (min since epoch).

DESCRIPTION:
(continued)

II. If the new epoch elements have not been found, then test to see if the D.C. integration must be initialized (SAVELEM $\neq$ 0). If so, the Adams-Bashforth subroutine must be initialized, then integration will begin at epoch. If not, the Adams-Bashforth buffer will be restored to the time of the last observation, then integration will continue until the elements are found for the new epoch time.

Having found the new epoch elements, the procedure will be the same as that listed under I.

This routine is called only by the ELMOUT subroutine when the elements are to be printed.

PURPOSE:	To print the weights and biases being used in the differential correction.
CALL SEQUENCE:	JMP PRINTW
INPUT:	Weights in WBUF Biases in BIBUF WANDBI = C/HLT, WBUF; C/HLT, BIBUF
OUTPUT:	Weights and biases printed on hard copy. (See description)
SUBROUTINES:	Program - PAGECON System - PANT, GLOP
STORAGE REQUIREMENTS:	92 Cells
DESCRIPTION:	This subroutine is called by option, if Column 3 on P Card 3 = 1. The weights and biases are printed as follows: (1) Print heading (2) Retrieve an entry from WBUF and BIBUF. Convert to output format. (3) Output the Sensor Number and the weights and biases. (4) When a new page is necessary, go to step (1). Otherwise, go to step (2). The subroutine will exit when a word (00000ZZZ) is found in WBUF.

PURPOSE: To set up the processed observation.

CALL SEQUENCE: JMP PROOBS

INPUT: Observation biases in BIBUF buffer, stored by station number.
Observations - start location is EBLOC; terminated by Z's.
Sensor file in SBLOC.
OSTROB = C/HLT, 0; C/HLT, N+1, N = number of observations.
SAVOBS = C/HLT, (Address), Address = 1st location to be used for a processed observation
PREDFLG = new epoch option

OUTPUT: Processed observation buffer, modified sensor buffer. (See description)

SUBROUTINES: Program - BIAS, SORTOB, SETSBUF, NXTOB, OBVEC, PAGECON, MOVBUF
System - GLOP

STORAGE REQUIREMENTS: 55 Cells

DESCRIPTION:

(1) Use BIAS subroutine to make a pass through the observations and apply biases.

(2) Call subroutine SORTOB to sort the observations in the order to be processed.

(3) Use subroutine SETSBUF to make a pass through the observations and move sensor information required to SBUF.

(4) Set OCOUNT, MCOUNT, OBSPROC, OBSREJ = 0/0
Set OBSLEFT = C/HLT, N; N = number of observations

(5) Retrieve an observation using NXTOB:

  (a) if"no sigmas"return, skip observation, print comment, subtract 1 from OBSLEFT and add 1 to OBSREJ.

  (b) if"no sensor"return, follow same procedure as (a).

DESCRIPTION:
(continued)

(6) Set the bits in OBFLG corresponding to the observed quantities:

1/1T47 - ρ

1/1T46 - A & h

1/1T45 - α & δ

1/1T44 - $\dot{\rho}$

(7) Use subroutine OBVEC to compute values and store them.

Continue steps (5)-(7) until an "end of observations return" from NXTOB; then

(8) If OCOUNT $\neq$ 0, use subroutine MOVBUF to move remaining observation buffer up.

(9) If PREDFLG = 2, set PRTIM = time of the last observation.

See the listed subroutines for output format and block formats.

PRNTMAT
SPIRDEC

PURPOSE:	To print the standard deviations and the correlation matrix of the delta elements.
CALL SEQUENCE:	JMP PRNTMAT
INPUT:	SCOUNTR = C/HLT, 0, C/HLT, N Where N = matrix size SIGN + 0 = $\sigma\ n$ 1 = $\sigma\ a_{xn}$ 2 = $\sigma\ a_{yn}$ 3 = $\sigma\ U_o$ 4 = $\sigma\ \Omega$ 5 = $\sigma\ i$ 6 = $\sigma\ B$ MATRIXB - MATRIXB + 27 Contains the correlation matrix. It is of variable size depending on the elements being corrected.
OUTPUT:	See description.
SUBROUTINES:	System - PANT, GLOP
STORAGE REQUIREMENTS:	74 Cells
DESCRIPTION:	Prints headings and values for the standard deviations. Then print the headings for the matrix and N lines of the matrix as specified by location SCOUNTR.

PURPOSE: To compute the perturbative acceleration due to direct solar radiation pressure, $\underline{\dot{r}}`_r$.

CALL SEQUENCE: JMP RDPRES

INPUT:

$$\text{RPFLAG} = \begin{cases} 0 & \text{no perturbations} \\ 1 & \text{compute perturbations} \end{cases}$$

RPCON3 = $\gamma \cdot B/2.2$

OUTPUT:

XRDGR = $\dot{x}`_r$

YRDGR = $\dot{y}`_r$

ZRDGR = $\dot{z}`_r$

SUBROUTINES: Philco - FSIN, FCOS
Program - ANGSUN

STORAGE REQUIREMENTS: 34 Cells

DESCRIPTION: If RPFLAG = 0, the subroutine sets XRDGR, YRDGR, and ZRDGR = F/O and exits. Otherwise it computes the following:

Call subroutine ANGSUN to compute $\underline{L}_\Theta$, then:

$$\cos\psi = \frac{\underline{L}_\Theta \cdot \underline{r}}{r}$$

If $\cos \psi \geq 0$, then satellite is in sunlight

$$\dot{x}`_r = (\nu \cdot B/2.2)\, L_{x\Theta}$$

$$\dot{y}`_r = (\gamma \cdot B/2.2)\, L_{y\Theta}$$

$$\dot{z}`_r = (\gamma \cdot B/2.2)\, L_{z\Theta}$$

If $\cos \psi < 0$, then compute:

$$\sin(\psi + \eta) = \sin\psi \cos\eta + \sin\eta \cos\psi$$

$$= (\cos^2\psi - 1)(1/r^2 - 1) + \cos\psi/r$$

If $\sin(\psi + \eta) \geq 0$, the satellite is illuminated so compute $\dot{r}`_r$. Otherwise $\dot{r}`_r$ is set = 0.

RDTSB
SPTRDEC

PURPOSE: To compute range rate.

CALL SEQUENCE: JMP RDTSB

INPUT:

XDOT = $\dot{x}$
YDOT = $\dot{y}$
ZDOT = $\dot{z}$
CXDOT = $\dot{X}$
CYDOT = $\dot{Y}$
CZDOT = $\dot{Z}$
XLX = L_x
XLY = L_y
XLZ = L_z

OUTPUT:

RHODT = $\dot{\rho}_c$
RODTX = $\dot{\rho}_x$
RODTY = $\dot{\rho}_y$
RODTZ = $\dot{\rho}_z$

SUBROUTINES: (None)

STORAGE REQUIREMENTS: 8 Cells

DESCRIPTION:

RODTX = XDOT + CXDOT = $\dot{\rho}_x = \dot{x} + \dot{X}$
RODTY = YDOT + CYDOT = $\dot{\rho}_y = \dot{y} + \dot{Y}$
RODTZ = ZDOT = $\dot{\rho}_z = \dot{z}$
RHODT = $\dot{\rho}_x L_x + \dot{\rho}_y L_y + \dot{\rho}_z L_z = \dot{\rho}_c$

PURPOSE: To read observations from logical tape 0.

CALL SEQUENCE: JMP READOBS

INPUT: OANDE = C/HLT, OBLOC; C/HLT, EBLOC
and observations on logical 0.

OUTPUT: Observations in EBLOC
SAVOBS - left address is the location of the Z's terminating the observations

SUBROUTINE: System - SYS, SYSNO, SYSIO

STORAGE REQUIREMENTS: 13 Cells

DESCRIPTION: This subroutine is called only when OUTOPT = 1. OUTOPT is the output option from the SPSJOB Card. The option should be set to 1 with more than 492 but not more than 984 observations in the input. In this case, the Executive routine will write all the observations on logical 0. SPIRDEC will read them in only once.

Subroutine READOBS does the following:

(1) Rewind logical 0
EBLOC ——>Index Register 3

(2) Set MCOUNT = 0
Read 1 block into the address at location T100; it is EBLOC initially.

(3) Test every 10th word for Z's in the block just read:

(a) if no Z's are found, add 10 to the address at TI00, add 10 to Index Register 3, add 1 to MCOUNT:

if MCOUNT = 12, go to step (2)
if MCOUNT < 12, continue searching for Z's.

(b) if Z's are found, save the location of Z's in the left address of SAVOBS and exit.

PURPOSE: To accept or reject a residual for range or angles for 1st pass.

CALL SEQUENCE:

```
TMA  (residual)
JMP  REJECT1
JAZ  (rejected)
```

INPUT:

ABSMX = rejection criteria (e.r.) for range and angles
SIGMAI = weight corresponding to the observed quantity

OUTPUT:

RCNT = C/HLT, N; C/HLT, 0
SUM = F.P.
REJCNT = C/HLT, N; C/HLT, 0

SUBROUTINES: Program: RESREJ1

STORAGE REQUIREMENTS: 6 Cells

DESCRIPTION: If the absolute value of the unweighted residual $\leq$ ABSMX, then RCNT is increased by 1, and the weighted residual squared is added to SUM.

If greater than ABSMX, then RCNT and SUM are not affected. But the REJCNT (rejected count) is increased by 1, the REJFLG is set to asterisks, and the (A) register is cleared to indicate rejection.

REJECT2
SPIRDEC

PURPOSE: To accept or reject a range rate residual for 1st pass.

CALL SEQUENCE:

```
TMA  (residual)
JMP  REJECT2
JAZ  (rejected)
```

INPUT:
ABMX2 = rejection criteria (e.r./kemin)
SIGMAI = σ_{0}

OUTPUT:
RCNT = C/HLT, N; C/HLT, 0
SUM = Floating point
REJCNT = C/HLT, N; C/HLT, 0

SUBROUTINES: Program: RESREJ1

STORAGE REQUIREMENTS: 6 Cells

DESCRIPTION: If |unweighted residual| ≤ ABMX2, then RCNT is increased by 1 and the weighted residual (squared) is added to SUM.

If |unweighted residual| > ABMX2, then RCNT and SUM are not affected, but the REJCNT (rejection count) is increased by 1, the REJFLG is set to astericks, and the (A) register is cleared to indicate rejection.

PURPOSE: To restore elements to continue integration.

CALL SEQUENCE: JMP RESICK

INPUT: BOO + 16 - BOO + 55
Contains the last 5 sets of elements

OUTPUT: ICK + 0 - ICK + 39
Contains the same element sets

SUBROUTINES: None

STORAGE REQUIREMENTS: 4 cells

DESCRIPTION: To continue integration for a prediction or the new epoch elements, the ICK buffer must be restored for the DIVDIF subroutine.

PURPOSE: To output observation residuals.

CALL SEQUENCE: JMP RESOUT1

INPUT:

OBFLG = flag for observed quantities

RHOC = ρ_c

RGSDL = $\Delta\rho$

ARSDL = ΔA

DRSDL = Δh

RRSDL = $\Delta\dot{\rho}$

OUTPUT: ρ, α & δ or A & h, $\dot{\rho}$ residuals

These residuals flagged with asterisk if rejected
Also, VMAG, DELU, U, BETA, Time, Station Number

SUBROUTINES:

Program - DELTAU, RESOUT2
System - GLOP, PAGECON

STORAGE REQUIREMENTS: 80 Cells

DESCRIPTION: If only range rate was observed, VMAG is not computed otherwise

$$\text{VMAG} = \sqrt{(\text{RGSDL})^2 + (\text{ARSDL})^2 + (\text{DRSDL})^2}$$

If angles were not observed, U, Δt, and BETA are not computed, otherwise:

$$\text{XOBS} = \rho_c L_x - X = x_o$$

$$\text{YOBS} = \rho_c L_y - Y = y_o$$

$$\text{ZOBS} = \rho_c L_z - Z = z_o$$

$$\text{OBSR} = \sqrt{x_o^2 + y_o^2 + z_o^2}$$

DESCRIPTION: (continued)

RESOUT1 uses subroutine DELTAU to compute Δu.

Then computes U:

$$U = \tan^{-1}\left[\frac{SINU}{COSU} + DELU\right]$$

Computes Δt (DELU)

$$\Delta t = (- \Delta u / \sqrt{\rho})\ (r^2 / k_e)$$

$$BETA = \sin^{-1}\left[(x_o W_x + y_o W_y + z_o W_z) / \sqrt{x_o^2 + y_o^2 + z_o^2}\right]$$

where $-90^\circ \leq BETA \leq 90^\circ$

Tests RESOPT:

if RESOPT = 0, then print angles in deg.

if RESOPT = 1, then print angles in km.

RESOUT2 &
RESOUT4
SPIRDEC

PURPOSE:	To convert time to BCD and output page heading if necessary.
CALL SEQUENCE:	JMP RESOUT2
INPUT:	T = t (Min.) STAID = 0 0XXX, X = Sta. No.
OUTPUT:	OBYEAR OBMO OBDAY } BCD time FOBHR FOBMIN FOBSEC STAID - Left Justified
SUBROUTINES:	Program - BCDTIM, PAGECON System - PANT
STORAGE REQUIREMENTS:	43 Cells
DESCRIPTION:	RESOUT2 converts time to BCD, tests if new page headings are needed. If so, it calls RESOUT4 to print page headings.

PURPOSE:	To accept or reject a residual on all D. C. passes except the first.
CALL SEQUENCE:	TMQ (weighted residual) TMA (residual) JMP RESREJ1 JAZ (residual rejected)
INPUT:	MAX = weighted rejection criteria ASTK = W/********
OUTPUT:	RCNT = C/HLT, N SUM = Floating point REJCNT = C/HLT, N REJFLG = * or blank
SUBROUTINES:	(None)
STORAGE REQUIREMENTS:	7 Cells
DESCRIPTION:	If $\mid$ weighted residual $\mid$ $\geq$ MAX, then REJCNT is increased by 1, REJFLG = *, and (A) reg. = 0 If $\mid$ weighted residual $\mid$ $<$ MAX, then RCNT is increased by 1, (weighted residual)^2 is added to SUM, and REJFLG = blank.

PURPOSE:	To restore values to the Adams-Bashforth buffer.
CALL SEQUENCE:	JMP RESW
INPUT:	QQ + 1 = L 2 = a_x 3 = a_y 4 = a_z 5 = h_x 6 = h_y 7 = h_z
OUTPUT:	W + 5 = L 6 = a_x 7 = a_y 8 = a_z 9 = h_x 10 = h_y 11 = h_z
SUBROUTINES:	Program - SETW
STORAGE REQUIREMENTS:	2 Cells
DESCRIPTION:	Restores values saved in QQ + 1 to QQ + 7 to W + 5 to W + 11 to continue the Adams-Bashforth integration.

RESWBF
SPIRDEC

PURPOSE:	To restore W buffer
CALL SEQUENCE:	JMP RESWBF
INPUT:	PREDBF buffer (11 cells)
OUTPUT:	W + 1 to W + 11 restored from PREDBF buffer (11 cells)
SUBROUTINES:	Program - SAVWBF
STORAGE REQUIREMENTS:	3 Cells
DESCRIPTION:	Restores W buffer to last point in ADBASH subroutine in order to restart integration.

PURPOSE: To update the revolution number.

CALL SEQUENCE: JMP REVSUB

INPUT:

UZ = U_z at t

OLDUZ = U_z at $t - \Delta t$

OUTPUT:

OLDUZ = U_z at t

REV = Rev No at t

SUBROUTINES: None

STORAGE REQUIREMENTS: 5 Cells

DESCRIPTION: A switch CNTSW must be preset before calling the subroutine:

If integrating forward from epoch:

CNTSW = C/JAN, CNTB; C/TMA, F/1

If integrating backward:

CNTSW = C/JAP, CNTB; C/TMA, F/-1

If the signs of UZ and OLDUZ are the same, the rev. no. is not modified.

If the signs are different, then a node has been crossed. If it is the ascending node, the rev. no. modified by 1.

PURPOSE: To compute range

CALL SEQUENCE: JMP RHOSB

INPUT:

X = x
Y = y
Z = z
CAPX = X
CAPY = Y
CAPZ = Z

OUTPUT:

RHOC = ρ_c
RHOX = ρ_x
RHOY = ρ_y
RHOZ = ρ_z

SUBROUTINES: Philco - FSQRT

STORAGE REQUIREMENTS: 11 Cells

DESCRIPTION:

RHOX = X+CAPX (ie. $\rho_x = x + X$)
RHOY = Y+CAPY
RHOZ = Z+CAPZ

RHOC = $\rho_c = \sqrt{(\text{RHOX})^2 + (\text{RHOY})^2 + (\text{RHOZ})^2}$

RINEL
SPIRDEC

PURPOSE: To restore initial elements to output cells.

CALL SEQUENCE: JMP RINEL

INPUT: INELT buffer

OUTPUT:

LPRINT = L (deg)
AXPRINT = a_{xno}
AYPRINT = a_{yno}
HXPRINT = h_x
HYPRINT = h_y
HZPRINT = h_z
BPRINT = $C_D A/m$
HSUBQP = H_q (km)
PAPRINT = period (min.)

SUBROUTINES: Program - SINEL

STORAGE REQUIREMENTS: 2 Cells

DESCRIPTION: Moves value from INELT buffer to output cells.

PURPOSE: To convert $\underline{r}$, $\dot{\underline{r}}$ elements to $\underline{a}$, $\underline{h}$, L

CALL SEQUENCE:

```
JMP  RR2AHL
(return)
```

INPUT:

X = x (km)
Y = y (km)
Z = z (km)
XDOT = $\dot{x}$ (km/sec)
YDOT = $\dot{y}$ (km/sec)
ZDOT = $\dot{z}$ (km/sec)

OUTPUT:

HXO = h_{xo}
HYO = h_{yo}
HZO = h_{zo}
AXNO = a_{xno}
AYNO = a_{yno}
XLO = L_{o}
+ cells on next page

SUBROUTINES:

Philco - FSQRT
System - ARCTAN

STORAGE REQUIREMENTS: 94 Cells

DESCRIPTION: Converts from $\underline{r}$, $\dot{\underline{r}}$ to $\underline{a}$, $\underline{h}$, L using the following formulation.
Called by subroutine INPUT only.

RR2AHL Formulation

$HXO = h_{xo} = y\,\dot{z} - z\,\dot{y}$

$HYO = h_{yo} = z\,\dot{x} - x\,\dot{z}$

$HZO = h_{zo} = x\,\dot{y} - y\,\dot{x}$

$P = h_{xo}^{2} + h_{yo}^{2} + h_{zo}^{2} = p$

$RTP = \sqrt{p}$

$WX = W_x = HXO/\sqrt{p}$ x → y, z

$R = \sqrt{x^2 + y^2 + z^2} = r$

$UX = U_x - x/r,$ x → y, z

$RDOT = \dot{r} = (x\dot{x} + y\dot{y} + z\dot{z})/r$

$ESINEV = \dot{r}\sqrt{p} = e \sin v$

$ECOSV = p/r - 1 = e \cos v$

$ESQ = (e \sin v)^2 + (e \cos v)^2 = e_o^2$

$EO = e_o = \sqrt{e_o^2}$

$VX = v_x = (rx - x\dot{r})$ ([illegible] $)x$ x → y, z

$A = a = p/1 - e^2$

$LLO = l_o = \tan^{-1} \frac{y - Vx}{x + Vy}$ $0 \leq l_o \leq 360$

$XNO = N = K_{\epsilon} / a^{3/2}$

$AX = a_x = U_x\, e \cos v - V_x\, e \sin v,$ x → y, z

$AXNO = a_{xno} = (a_y\ W_x - a_x\ W_y) / \sqrt{1 - W_z^2}$

$AYNO = a_z \sqrt{1 - Wz^2} - Wz\ (a_x\ W_x + a_y\ W_y) / \sqrt{1 - W_z^2}$

$$v - E = \tan^{-1} \left[\frac{e \sin v\,(1 + \sqrt{1 - e^2} + e \cos v)}{(e \sin v)^2 + (e \cos v + 1)\ (1 + \sqrt{1 - e^2})} \right]$$

$QQ + 2 = v - M = v - E + r\,\epsilon \sin v\ \sqrt{1 - e^2} / p$

$XLO = L_o = l_o - (v - M)$

SAVCON
SPIRDEC

PURPOSE:	To save card images of the parameter cards.
CALL SEQUENCE:	TMD C/HLT, SCONBUF+X JMP SAVCON
INPUT:	Index register 3 = 1st location of a parameter card image in CONBUF
OUTPUT:	See description.
SUBROUTINES:	None
STORAGE REQUIREMENTS:	4 Cells
DESCRIPTION:	The parameter card image is moved to the buffer, SCONBUF, to be used by subroutine ELMOUT for output purposes.

SCONBUF + 0 to + 9	P Card 2
+ 10 to + 19	P Card 3
+ 20 to + 29	P Card 5
+ 30 to + 39	P Card 6
+ 40 to + 49	P Card 7
+ 50 to + 59	P Card 8

PURPOSE: To save the values necessary to continue integration to the new epoch elements.

CALL SEQUENCE: JMP SAVELM

INPUT: 5 sets of elements in the ICK buffer
REV = revolution number
OLDUZ = u_z at the previous time point
W - ADBASH buffer

OUTPUT: See description

SUBROUTINES: Program - SAVWBF, SAVICK

STORAGE REQUIREMENTS: 5 Cells

DESCRIPTION: This subroutine is only called by the D.C. control subroutine (CNTRL) if the new epoch time requested is outside the span of the observations and on the opposite side of the old epoch from the last observation. A third requirement for using this routine is that the old epoch must be within the span of the observations.

The subroutine will:

(1) Use SAVWBF to save W + 1 → W + 11

(2) Use SAVICK to save ICK → ICK + 39

(3) Save REV in SREV
u_z in SOLDUZ

(4) Set SAVELM = 0 indicating this subroutine was used

These values are used in the subroutine [illegible]

PURPOSE: To save values to compute the new epoch elements.

CALL SEQUENCE:

```
TMD  C/HLT, ICK+X; C/HLT, BOO+8
JMP  SAVICK
```

INPUT:

AXN = a_{xn}

AYN = a_{yn}

B = C_D A/m

A = a

ESQ = e^2

SINI = sin i

P = p

XN = n

ICK buffer from DIVDIF subroutine

OUTPUT:

BOO + 0 = a_{xn}

+ 1 = a_{yn}

+ 2 = B

+ 3 = a

+ 4 = e^2

+ 5 = sin i

+ 6 = p

+ 7 = n

+ 8 thru +15 = new epoch elements

+ 16 thru +55 = last 5 sets of integrated elements

SUBROUTINES: Program: SAVW, SETW, SUBXYZ, RESW

STORAGE REQUIREMENTS: 14 Cells

DESCRIPTION: Saves the new epoch elements in BOO + 8 thru BOO + 15. Values saved in BOO + 0 thru BOO + 7 are necessary to prepare the new epoch elements for output. In BOO + 16 thru BOO + 55 the last 5 sets of elements are saved to continue integration if a prediction is desired.

PURPOSE: To save the elements in the Adams-Bashforth buffer.

CALL SEQUENCE: JMP SAVW

INPUT:

W + 5 = L
6 = a_x
7 = a_y
8 = a_z
9 = h_x
10 = h_y
11 = h_z

OUTPUT:

QQ + 1 = L
2 = a_x
3 = a_y
4 = a_z
5 = h_x
6 = h_y
7 = h_z

SUBROUTINES: Program - SETW

STORAGE REQUIREMENTS: 2 Cells

DESCRIPTION: Moves values from W + 5 thru W + 11 to QQ + 1 thru QQ + 7 so that other values may be placed in W + 5 thru W + 11 to be able to use the SUBXYZ subroutine.

SAVWBF
SPIRDEC

PURPOSE:	To save W buffer
CALL SEQUENCE:	JMP SAVWBF
INPUT:	W buffer
OUTPUT:	W + 1 to W + 11 → PREDBF +0 to PREDBF + 10
SUBROUTINES:	(None)
STORAGE REQUIREMENTS:	5 Cells
DESCRIPTION:	Saves 11 cells of the W buffer in order to restart ADBASH subroutine.

PURPOSE: To save elements for interpolation.

CALL SEQUENCE: JMP SAV5PTS

INPUT:

$$\begin{aligned} W + 1 &= t \quad (\text{min}) \\ 5 &= L \\ 6 &= a_x \\ 7 &= a_y \\ 8 &= a_z \\ 9 &= h_x \\ 10 &= h_y \\ 11 &= h_z \end{aligned}$$

OUTPUT: See description.

$$\left.\begin{aligned} ICK + 0 &= t \\ + 1 &= L \\ + 2 &= a_x \\ + 3 &= a_y \\ + 4 &= a_z \\ + 5 &= h_x \\ + 6 &= h_y \\ + 7 &= h_z \end{aligned}\right\} \text{5 sets}$$

$$\downarrow$$

$$+ 39$$

SUBROUTINES: None

STORAGE REQUIREMENTS: 8 Cells

DESCRIPTION: Buffer ICK contains 5 sets of elements at $t_1 - t_5$. When t_1 is no longer needed, elements at $t_2 - t_5$ are moved to replaced $t_1 - t_4$. And the elements at t_6 in the W buffer are moved to replace the elements at t_5.

PURPOSE: To compute θ, sin θ, cos θ

CALL SEQUENCE:

```
TMA  T
JMP  SENLOC
```

INPUT:

T = t = minutes since epoch

XLAMBA = λ_E

THGR = θ_{gr}

RPTIM = .0043752691

OUTPUT:

THTA = θ at time t

SINTH = sin θ

COSTH = cos θ

SUBROUTINES: Philco - FCOS, FSIN

STORAGE REQUIREMENTS: 6 Cells

DESCRIPTION: THTA = θ = $t\,(.0043752691) + \lambda_E + \theta_{gr}$

PURPOSE:	To set up a modified SBLOC in SBUF.
CALL SEQUENCE:	JMP SETSBUF
INPUT:	BIASAD = C/HLT, EBLOC; C/HLT, BIBUF Observations starting in EBLOC Sensors in SBLOC
OUTPUT:	Modified SBLOC in buffer SBUF See buffer layout under description.
SUBROUTINE:	System - SGET
STORAGE REQUIREMENTS:	19 Cells
DESCRIPTION:	Starting with the first observation, extract the sensor number:

(1) if the sensor information is in SBUF, go to the next observation

(2) if the sensor is not in SBUF, call subroutine SGET to unpack the entry in SBLOC, and store only 5 quantities for the sensor in SBUF, then go to the next observation

(3) if SBUF is full (30 sensors), exit without error indication. The retrieval routine GETSEN has an error exit.

The sensor information must be moved, since SBLOC is used for a working buffer and for temporary storage. SBUF is the same buffer as BIBUF; BIBUF (bias buffer) is no longer needed at this point.

DESCRIPTION:
(continued)

SBUF Format

SBUF + 0	00000SSS	Sensor Number
1	ϕ (radians)	PHIRD
2	λ (radians)	XLAMBA
3	$x / \cos \theta$	XOVCT
4	Z	CAPZ
⋮		
145	00000SSS	
146	ϕ	
147	λ	
148	$x / \cos \theta$	
149	Z	
150	00000ZZZ	

5 words / entry - to a maximum of 30 sensors - terminated by 00000ZZZ after the last entry.

PURPOSE: To move interpolated elements at time t to the Adams-Bashforth buffer.

CALL SEQUENCE: JMP SETW

INPUT:

ICK + 40 = L
41 = a_x
42 = a_y
43 = a_z
44 = h_x
45 = h_y
46 = h_z

OUTPUT:

W + 5 = L
6 = a_x
7 = a_y
8 = a_z
9 = h_x
10 = h_y
11 = h_z

SUBROUTINES: None.

STORAGE REQUIREMENTS: 4 Cells

DESCRIPTION: ICK + 40 thru ICK + 47 contains elements at time t computed by the DIVDIF subroutine. To compute the x, y, z position at this time, these values must be in W + 5 thru W + 11 to use the SUBXYZ subroutine.

PURPOSE:	To save initial elements for output.
CALL SEQUENCE:	JMP SINEL
INPUT:	LPRINT = L_o (deg) AXPRINT = a_{xno} AYPRINT = a_{yno} HXPRINT = h_{xo} HYPRINT = h_{yo} HZPRINT = h_{zo} BPRINT = $C_D A/m$ HSUBQP = perigee altitude (km) PAPRINT = period (minutes)
OUTPUT:	Input values in buffer INELT - INELT+8
SUBROUTINES:	None
STORAGE REQUIREMENTS:	4 Cells
DESCRIPTION:	Moves input values to the INELT buffer in the order indicated.

PURPOSE: To sort the observations by time.

CALL SEQUENCE: JMP SORTOB

INPUT: MCOUNT = C/HLT, 0; C/HLT, N
N = number of negative observation times

OANDE = C/HLT, OBLOC; C/HLT, EBLOC

OUTPUT: See description

SUBROUTINES: None

STORAGE REQUIREMENTS: 33 Cells

DESCRIPTION:

(1) Set the right address of TEMP1 and TEMP2 to the number of observations minus one.
Set switch SORTOB3 to sort the observations in true time order.

(2) EBLOC → Index Register 4
EBLOC + 10 → Index Register 5
C/HLT, OBLOC; C/HLT, EBLOC → QQ

(3) PREDBF → Index Register 6

(4) Sort observations in time order - make as many passes through the observations as the number in TEMP1, using TEMP2 to tell when one pass is complete. TEMP2 will match TEMP1 at the start of each pass.

(5) Test MCOUNT:

(a) if = 0, exit

(b) if ≠ 0, then there are some observations before epoch, (i.e. some negative times). These must be resorted in reverse order.

Example: buffer order would be
-2, ... -100, 2 100
since observations will be
retrieved in this order.

PURPOSE: To output the prediction ephemeris.

CALL SEQUENCE: JMP SUBOUT

INPUT:

NEWPAGE = page indicator

PFLAG = print option

X = x

Y = y

Z = z

XDOT = $\dot{x}$

YDOT = $\dot{y}$

ZDOT = $\dot{z}$

T = t (min)

$$\left.\begin{array}{l}\text{HEDLIN}\\ \text{HEDLINI}\\ \text{VALLIN}\\ \text{VALLINI}\end{array}\right\}\text{ from SUBOUTI}$$

OUTPUT: According to value of PFLAG. (See description.)

SUBROUTINES: System - GLOP, PANT

Program - PAGECON, PHLAH, BCDTIM

STORAGE REQUIREMENTS: 65 cells

DESCRIPTION: If NEWPAGE = 0, then outputs headings at the top of a new page, using locations HEDLIN and HEDLINI.

If PFLAG = 2 or 3, subroutine PHLAH is called to compute Ø, λ, h.

If PFLAG = 1 or 3, then x, y, z, $\dot{x}$, $\dot{y}$, $\dot{z}$, are converted to output units of km and km/sec.

The values requested by PFLAG are then printed and PAGECON is called to update the line count.

PURPOSE: To initialize subroutine SUBOUT.

CALL SEQUENCE: JMP SUBOUTI

INPUT: PFLAG = 0, 1, 2, or 3 T47 (print option from P card 4)

OUTPUT: See description.

SUBROUTINES: Program - PAGECON

STORAGE REQUIREMENTS: 14 cells

DESCRIPTION: Sets up parameters for SUBOUT (HEDLIN, HEDLINI, VALLIN, VALLINI) according to PFLAG.

Then use PAGECON to force a page and sets NEWPAGE = 0.

If PFLAG = 0 or 2:

HEDLIN = C/TMA, LINE4; C/TIJ, LINE4B
HEDLINI = 0 —— 0
VALLIN = C/TMA, POC7D; C/TIJ, HEDL23
VALLINI = 0 —— 0

If PFLAG = 1

HEDLIN = C/TMA, LINE4; C/TIJ, LINE4A
HEDLINI = 0 —— 0
VALLIN = C/TMA, POC7D; C/TIJ, HEDL21
VALLINI = 0 —— 0

And sets switch SUBOUT2 to jump to SUBOUT7.

If PFLAG = 3

HEDLIN = C/TMA, LINE4; C/TIJ, LINE4B
HEDLINI = C/HLT, 0; C/TIJ, LINE4A
VALLIN = C/TMA, POC7D; C/TIJ, HEDL23
VALLINI = C/HLT, 0; C/TIJ, HEDL21

And sets switch SUBOUT2 to jump to SUBOUT8.

PURPOSE: To compute $\underline{r}$, $\dot{\underline{r}}$ from $\underline{A}$, $\underline{h}$, L

CALL SEQUENCE:

```
JMP    SUBXYZ
JMP    (ERROR)
JMP    NORMAL
```

INPUT:

$W + 5 = L$

$W + 6 = a_x$

$W + 7 = a_y$

$W + 8 = a_z$

$W + 9 = h_x$

$W + 10 = h_y$

$W + 11 = h_z$

OUTPUT:

X = x
Y = y
Z = z
XDOT = $\dot{x}$
YDOT = $\dot{y}$
ZDOT = $\dot{z}$

plus necessary intermediate quantities listed on following pages.

SUBROUTINES: Program - ARCTAN

STORAGE REQUIREMENTS: 100 cells

Kepler's Equation:
(1)

$$COSEO = \cos(E + \omega)$$

$$SINEO = \sin(E + \omega)$$

$$ECOSE = SINEO\ (a_{yn}) + COSEO\ (a_{xn})$$

$$QQ = 1 - ECOSE$$

$$ESINE = SINEO\ (a_{xn}) - COSEO\ (a_{yn})$$

Test:

$$EPSI = \left| \frac{ESINE + U - EOI}{1 - ECOSE} \right|$$

if $< \epsilon$, iterate again to Eq. (1) after replacing E01 by E01 - EPSI to a maximum of 50 times, then take error exit from subroutine

if $\geq \epsilon$, continue

$$R = A\ (1 - ECOSE) = r$$

$$RDOT = (\sqrt{a}\sqrt{\mu}\ / r)\ ESINE = \dot{r}$$

$$RVDOT = (\sqrt{a}\sqrt{\mu}\ / r)\ \sqrt{1-e^2} = r\dot{v}$$

$$COSU = ESINE\left[(1- \sqrt{1-e^2}\quad a_{yn} - a_{xn} + COSEO\quad a/r\right.$$

$$SINU = SINEO - a_{yn} - a_{xn}\ ESINE\ (1- \sqrt{1-e^{2-1}})\quad a/r$$

$$AR = a/r$$

$$UX = COSU \cdot N_x + SINU \cdot M_x$$

$$UY = COSU \cdot N_y + SINU \cdot M_y$$

$$UZ = COSU \cdot N_z + SINU \cdot M_z$$

$$VX = COSU \cdot M_x - SINU \cdot N_x$$

$$VY = COSU \cdot M_y - SINU \cdot N_y$$

$$VZ = COSU \cdot M_z - SINU \cdot N_z$$

$X = r (U_x) = x$

$Y = r (U_y) = y$

$Z = r (U_z) = z$

$XDOT = \dot{r} U_x + r \dot{v} V_x = \dot{x}$

$YDOT = \dot{r} U_y + r \dot{v} V_y = \dot{y}$

$ZDOT = \dot{r} U_z + r \dot{v} V_z = \dot{z}$

PURPOSE:	To write a binary ephemeris tape on logical
CALL SEQUENCE:	JMF TAPEW
INPUT:	W+1 = t (min) X (e.r.) Y (e.r.) Z (e.r.) XDOT (e.r./kemin) YDOT (e.r./kemin) ZDOT (e.r./kemin) TAPCNT = C/HLT, TAPBUF + X
OUTPUT:	One block on the binary ephemeris tape for XYZLA subroutine when buffer is full.
SUBROUTINES:	System - SYS, SYSNO, SYSIO
STORAGE REQUIREMENTS:	16 Cells
DESCRIPTION:	Adds t, x, y, z, $\dot{x}$, $\dot{y}$, $\dot{z}$ to tape output buffer (TAPBUF). When the buffer is full, one block (128 words) is written on tape. TAPCNT is updated.

PURPOSE:	To compute temperature at epoch.
CALL SEQUENCE:	JMP TEMP
INPUT:	F10AV = $\bar{F}_{10}$ F10 = F_{10} AP = A_p ORGDAY = days from beginning of year to epoch = D
OUTPUT:	TEMPO = T_o (°K)
SUBROUTINES:	None
STORAGE REQUIREMENTS:	19 cells
DESCRIPTION:	$\bar{T}_o = 974^\circ + 4.2^\circ (\bar{F}_{10} - 150) + 0.004^\circ (\bar{F}_{10} - 150)^2$ $T_o' = \bar{T}_o + 1.9^\circ (F_{10} - \bar{F}_{10})$ $T_o = T_o' + \left[\bar{F}_{10} \sin\left\{4\pi \frac{(D-60)}{365}\right\}\right]\left[0.39^\circ + 0.15^\circ \sin\left\{2\pi \frac{(D-150}{365}\right\}\right]$

THGRC
SPIRDEC

PURPOSE: To compute θ at epoch

CALL SEQUENCE: JMP THGRC

INPUT:
TOY = 0————0Y, Y = BCD year
ORGTM = fraction of epoch day
THGRO = θ_{gr} from TLC subroutine
ORGDAY = days since beginning of year

OUTPUT:
TO = days and fraction since beginning of year
THGR = θ_{t_0}
ORGTM = minutes in epoch day

SUBROUTINES: Program - TLC, MOD2PI

STORAGE REQUIREMENTS: 13 Cells

DESCRIPTION: THGR = θ_{gr}+ (Days since beginning of year) (0.9856473354) + (fraction of day) (360.9856473)

Converts ORGTM from fraction of day to minutes

PURPOSE: To read the weight tape and set up the weight and bias buffers.

CALL SEQUENCE: JMP WEIGHT

INPUT: Weights and biases on logical tape 7
WANDBI = C/HLT, WBUF; C/HLT, BIBUF
NXTWGTA = C/HLT, ABUF + 120; C/HLT, 0

OUTPUT: See description

SUBROUTINES: System - PANT, SYS, SYSNO, SYSIO
Program - XSRCH

STORAGE REQUIREMENTS: 82 Cells

DESCRIPTION: This subroutine is called if WGTFLG = 1 (Column 2 on P Card 3), which indicates that a weight tape is mounted and should be read.

The following procedure is executed:

(1) Read a block from logical tape 7 initially, then only read a new block when all cards in the block have been processed:

- (a) if the first word of a card is ENDSIGMA, the tape is complete, rewind logical 7 and exit.
- (b) if not ENDSIGMA, unpack one card using XSRCH
  - (1) Set WGTBUF = 00000SSS where SSS = Sensor Number
  - (2) Save the address of the next card in NXTWGTA.

DESCRIPTION:
(continued)

(2) Search WBUF to find a word of 00000ZZZ or the same Sensor Number.

(a) if the Sensor Number is found, replace the assembled values with the new values after conversion as follows:

Word	WBUF (Weight Buffer)	Word	BIBUF (Bias Buffer)
0	00000SSS	0	00000SSS
1	$1/\sigma_\rho$ (e.r.)	1	$-B_\rho$ (e.r.)
2	$1/\sigma_A$ (rad)	2	$-B_A$ (rad)
3	$1/\sigma_h$ (rad)	3	$-B_h$ (rad)
4	$1/\sigma_\rho$ (e.r/K)	4	$-B_\rho$ (e.r/K)
		5	$-B_\tau$ (min)

Go to Step (1).

(b) if the sensor number is not found, but 00000ZZZ is found check to see if there is room for another entry

(1) if there is no room (i.e. 30 sensors have been entered), print a comment and go to step (1), skipping the entry

(2) if there is room, follow same procedure of storing and computing values at (2)(a), also add a word of 00000ZZZ at the end of each buffer.

PURPOSE:	To print the input observations.
CALL SEQUENCE:	JMP WILBUR
INPUT:	Observations starting in location EBLOC, terminated by a word of Z's.
OUTPUT:	Observations printed in card format on hard copy.
SUBROUTINES:	Program - PAGECON, SEPSUB System - PANT, GLOP, DKLOK
STORAGE REQUIREMENTS:	164 Cells
DESCRIPTION:	This subroutine is called by option, if Column 1 on P Card 3 = 1. The observations are in OBLOC format and must be converted to card format for output. The observations will be printed in the same order as they are in the input deck, as they have not yet been sorted. In addition, the column numbers will be printed at the top of each page. A space will appear between each field of the observation card for clarity.

WSETUP
SPIRDEC

PURPOSE: To set up W buffer for Adams-Bashforth integration.

CALL SEQUENCE: JMP WSETUP

INPUT:

XLO = L_o
AXO = a_{xo}
AYO = a_{yo}
AZO = a_{zo}
HXO = h_{xo}
HYO = h_{yo}
HZO = h_{zo}

OUTPUT:

W + 1 = F/O
W + 5 = XLO
W + 6 = AXO
W + 7 = AYO
W + 8 = AZO
W + 9 = HXO
W +10 = HYO
W +11 = HZO

SUBROUTINES: (None)

STORAGE REQUIREMENTS: 10 Cells

DESCRIPTION: Moves element set at epoch, i.e. (W + 1 = T) to W buffer for subroutine ADBASH.

APPENDIX

ADAMS-BASHFORTH INTEGRATION

The Adams-Bashforth (A-B) method together with the Runge-Kutta (R-K) method of integration are employed in this subroutine. R-K is used as a starting procedure so that an adequate number of step-wise solutions may be obtained to build a difference table needed by the A-B method. The interval of integration is automatically varied to keep the discrepancy between the integrated values and an absolute error check within prescribed limits.

I-1. R-K PROCEDURE FOR STARTING

To start the Adams-Bashforth procedure, a series of R-K integration steps (at equal intervals of X) are generated until enough points are available to set up the difference table needed by A-B.

The system of equations to be solved is given by.

$$Y_i' = f_i\ (X, Y_1, Y_2, \ldots, Y_7) \qquad i = 1, 2, \ldots, 7$$

$$Y_i\ (X_o) = Y_{io}$$

Let Y be the value of Y at $X=X_n$ and f_n the derivative of Y at $X=X_n$, and let h be the step size of the independent variable X (note that the subscript "i" has been omitted for simplicity). The R-K method uses the classical fourth-order formulas:

$$K_1 = hf\ (X_n, Y_n)$$

$$K_2 = hf\ (X_n + \frac{1}{2} h, Y_n + \frac{1}{2} K_1)$$

$$K_3 = hf\,(X_n + \frac{1}{2}h, Y_n + \frac{1}{2}K_2)$$

$$K_4 = hf\,(X_n + h, Y_n + K_3)$$

$$Y_{n+1} = Y_n + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4)$$

a. Interval Control and Modifications

The method of interval control, while in the R-K section, consists of generating, from the point X_n, Y_n and its derivative f_n, two R-K points Y_{n+1}, Y_{n+2} and their derivatives f_{n+1}, f_{n+2}. With this information, a Simpson's Rule integration is performed on the integral:

$$\Delta Y = \int_{j=n}^{n+2} f_j \, dx$$

By Simpson's Rule:

$$\Delta Y = \frac{h}{3}(f_n + 4\,f_{n+1} + f_{n+2})$$

The following evaluation is made for each of the seven equations:

$$\delta = \left|\Delta Y - (Y_{n+2} - Y_n)\right| - \epsilon$$

ϵ_i = absolute error control supplied by the user for each equation.

If, for any of the seven equations, the value δ equals or exceeds 0, then h is reduced by $10^{-0.2}$, and the integration is restarted from X_n, Y_n. This procedure is repeated until all δ are less than 0; then the values of X_{n+2}, Y_{n+2} are returned to the user as valid points.

b. Difference Table Construction

After the appropriate number* of R-K integration points have been generated, and seven values of f_{n+j} (j = 0, 1, ..., 6) at equal steps are available, then the difference table required by A-B is constructed.

---

*The program generates 4 R-K points per 1 A-B point, therefore, 25 R-K points must be available to start A-B.

The difference table is formed as follows:

$$
\begin{array}{ccccccc}
\nabla^{0}_{n} & & & & & & \\
 & \nabla^{1}_{n+1} & & & & & \\
\nabla^{0}_{n+1} & & \nabla^{2}_{n+2} & & & & \\
 & \nabla^{1}_{n+2} & & \nabla^{3}_{n+3} & & & \\
\nabla^{0}_{n+2} & & \nabla^{2}_{n+3} & & \nabla^{4}_{n+4} & & \\
 & \nabla^{1}_{n+3} & & \nabla^{3}_{n+4} & & \nabla^{5}_{n+5} & \\
\nabla^{0}_{n+3} & & \nabla^{2}_{n+4} & & \nabla^{4}_{n+5} & & \nabla^{6}_{n+6} \\
 & \nabla^{1}_{n+4} & & \nabla^{3}_{n+5} & & \nabla^{5}_{n+6} & \\
\nabla^{0}_{n+4} & & \nabla^{2}_{n+5} & & \nabla^{4}_{n+6} & & \\
 & \nabla^{1}_{n+5} & & \nabla^{3}_{n+6} & & & \\
\nabla^{0}_{n+5} & & \nabla^{2}_{n+6} & & & & \\
 & \nabla^{1}_{n+6} & & & & & \\
\nabla^{0}_{n+6} & & & & & &
\end{array}
$$

where:

$$\nabla^{r}_{k} = \nabla^{r}_{k} - \nabla^{r}_{k-1}$$

$$\nabla^{0}_{k} = f_{k} \; ; \; k = n, \; n+1, \; \ldots \; n+6$$

After this difference table has been formed, only the ∇^{r}_{n+6} differences are saved for A-B computations.

[illegible]-2. A-B SIXTH ORDER INTEGRATION

The A-B integration method consists of:

1. Computing a predictor
2. Computing new differences, ∇^{r}, based on the results of the predictor.
3. Computing the corrector (the final result).

The predictor is of the form

$$Y_{n+1} = Y_n + h \sum_{r=0}^{6} B_r \nabla_n^r$$

$B_0 = 1$

$B_1 = 1/2$

$B_2 = 5/12$

$B_3 = 3/8$

$B_4 = 251/720$

$B_5 = 95/288$

$B_6 = 19087/60480$

The corrector is of the form

$$Y_{n+1} = Y_n + h \sum_{r=0}^{6} B_r' \nabla_{n+1}^r$$

$B_0' = 1$

$B_1' = -1/2$

$B_2' = -1/12$

$B_3' = -1/24$

$B_4' = -19/720$

$B_5' = -3/160$

$B_6' = -863/60480$

The differences ∇^{r}_{n+1} used in the corrector are computed by evaluating the derivative of the predictor f_{n+1} (∇^{0}_{n+1}) ; then

$$\nabla^{r}_{n+1} = \nabla^{r-1}_{n+1} - \nabla^{r-1}_{n} \quad ; \quad r=1, 2, \ldots, 6$$

-3. A-B INTERVAL CONTROL AND MODIFICATION

After the corrector has been computed, the quantity δ is evaluated.

$$\delta = \left|\text{predictor - corrector}\right| - \epsilon$$

ϵ_i = absolute error control

If, for any equation, $\delta>0$; then the interval, h, will be decreased by 1/2. New differences, ∇^r (r=1, 2, ...6), based on half the old interval are then computed by:

$$\begin{bmatrix} \nabla_n^{1} \\ \nabla_n^{2} \\ \nabla_n^{3} \\ \nabla_n^{4} \\ \nabla_n^{5} \\ \nabla_n^{6} \end{bmatrix} = \begin{bmatrix} .5 & .125 & .0625 & .0390625 & .02734375 & .0205078125 \\ 0 & .25 & .125 & .078125 & .0546875 & .041015625 \\ 0 & 0 & .125 & .09375 & .0703125 & .0546875 \\ 0 & 0 & 0 & .0625 & .0625 & .0546875 \\ 0 & 0 & 0 & 0 & .03125 & .0390625 \\ 0 & 0 & 0 & 0 & 0 & .015625 \end{bmatrix} \begin{bmatrix} \nabla_n^{1} \\ \nabla_n^{2} \\ \nabla_n^{3} \\ \nabla_n^{4} \\ \nabla_n^{5} \\ \nabla_n^{6} \end{bmatrix}$$

$$\nabla_n^{0} = \nabla_n^{0}$$

With the new h and ∇^{r}_{n}, repeated attempts are made to integrate from x to x+h until, for all equations, $\delta \leq 0$.

When all $\delta \leq 0$, a new set of differences (∇^{r}_{n+1}) will be computed by evaluating the derivative of the corrector (∇^{0}_{n+1}), then evaluating:

$$\nabla^{r}_{n+1} = \nabla^{r-1}_{n+1} - \nabla^{r-1}_{n} \quad ; \quad r=1, 2, \ldots, 6$$

The A-B routine will check if a larger interval h may be used and still keep the error within prescribed limits. The doubling criteria are.

1. There must have previously been seven integrations by AB without a change in h.

2. All $\delta \leq -.996e$

If these criteria are satisfied, then h is doubled and a new set of differences $\bar{\nabla}^{r}_{n+1}$ based on twice the old interval are computed by:

$$\begin{bmatrix} \bar{\nabla}^{1}_{n+1} \\ \bar{\nabla}^{2}_{n+1} \\ \bar{\nabla}^{3}_{n+1} \\ \bar{\nabla}^{4}_{n+1} \\ \bar{\nabla}^{5}_{n+1} \\ \bar{\nabla}^{6}_{n+1} \end{bmatrix} = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 4 & -4 & 1 & 0 & 0 \\ 0 & 0 & 8 & -12 & 6 & -1 \\ 0 & 0 & 0 & 16 & -32 & 24 \\ 0 & 0 & 0 & 0 & 32 & -80 \\ 0 & 0 & 0 & 0 & 0 & 64 \end{bmatrix} \begin{bmatrix} \nabla^{1}_{n+1} \\ \nabla^{2}_{n+1} \\ \nabla^{3}_{n+1} \\ \nabla^{4}_{n+1} \\ \nabla^{5}_{n+1} \\ \nabla^{6}_{n+1} \end{bmatrix}$$

$$\bar{\nabla}^{0}_{n+1} = \nabla^{0}_{n+1}$$

The Y_{n+1} are returned to the user at this point. The new $\bar{\nabla}^{r}_{n+1}$ and h (if any) will be tried on the next step.

4.SPECIAL CONSIDERATION FOR $Y_{1,n}$

The first equation, $Y_{1,n}$ (located at W+5), is carried modulo 2π for all computations within ADBASH. This feature allows absolute (only) error control to be used for this variable. The value $Y_{1,n+1}$, however, will not be modulo 2π; but will be $Y_{1,n}$ (as entered by the user) plus $\Delta Y_{1,n}$ by integration.

The "modulo 2π" feature restricts the user to input Y in radians; and also restricts the derivative routine to accept $Y_{1,n}$ in radians (mod 2π).

I-5. FLOW CHARTS AND WORK REGION DESCRIPTION

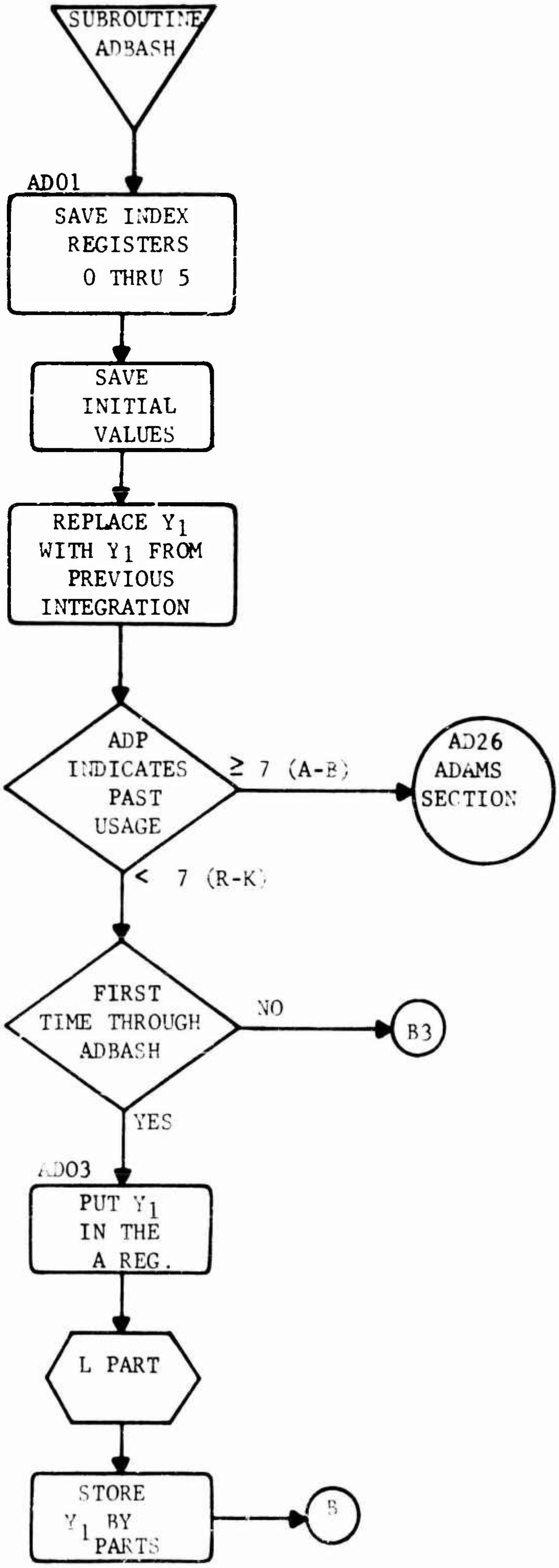
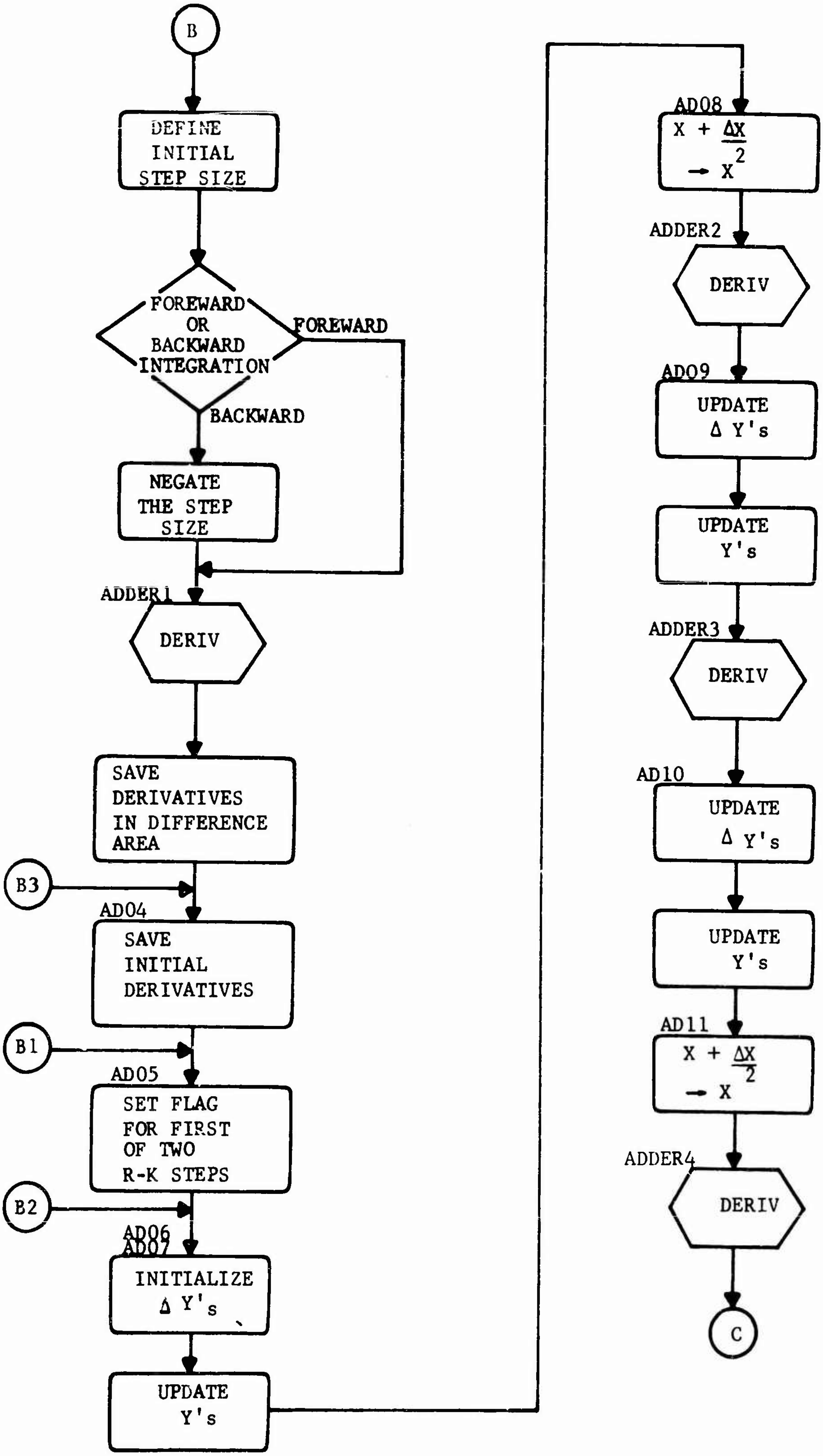
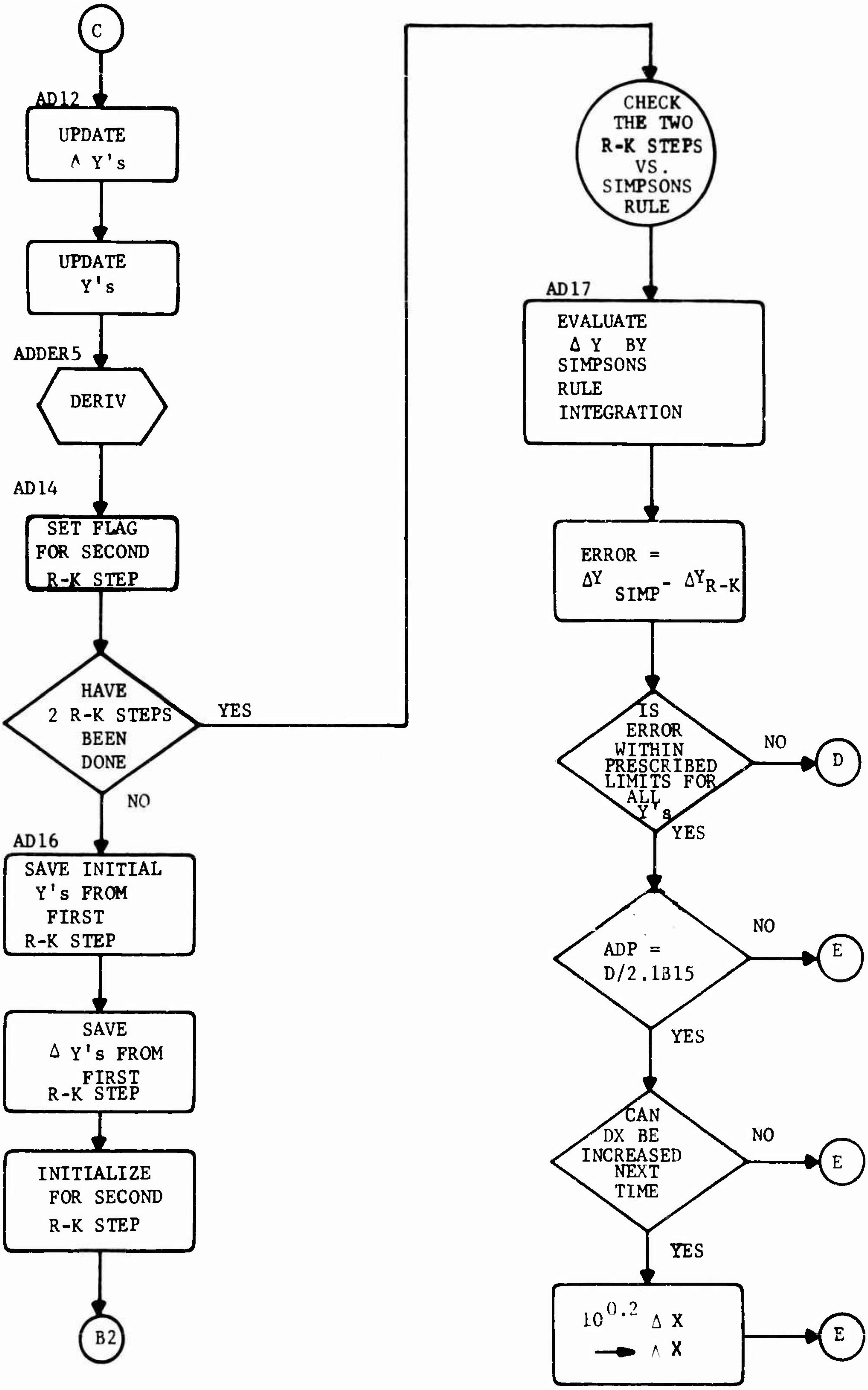


FIGURE I-1. ADAMS-BASHFORTH FLOW DIAGRAM (1 OF 6)



.FIGURE I-1. ADAMS-BASHFORTH FLOW DIAGRAM (2 of 6)



.FIGURE I-1. ADAMS-BASHFORTH FLOW DIAGRAM (3 of 6)

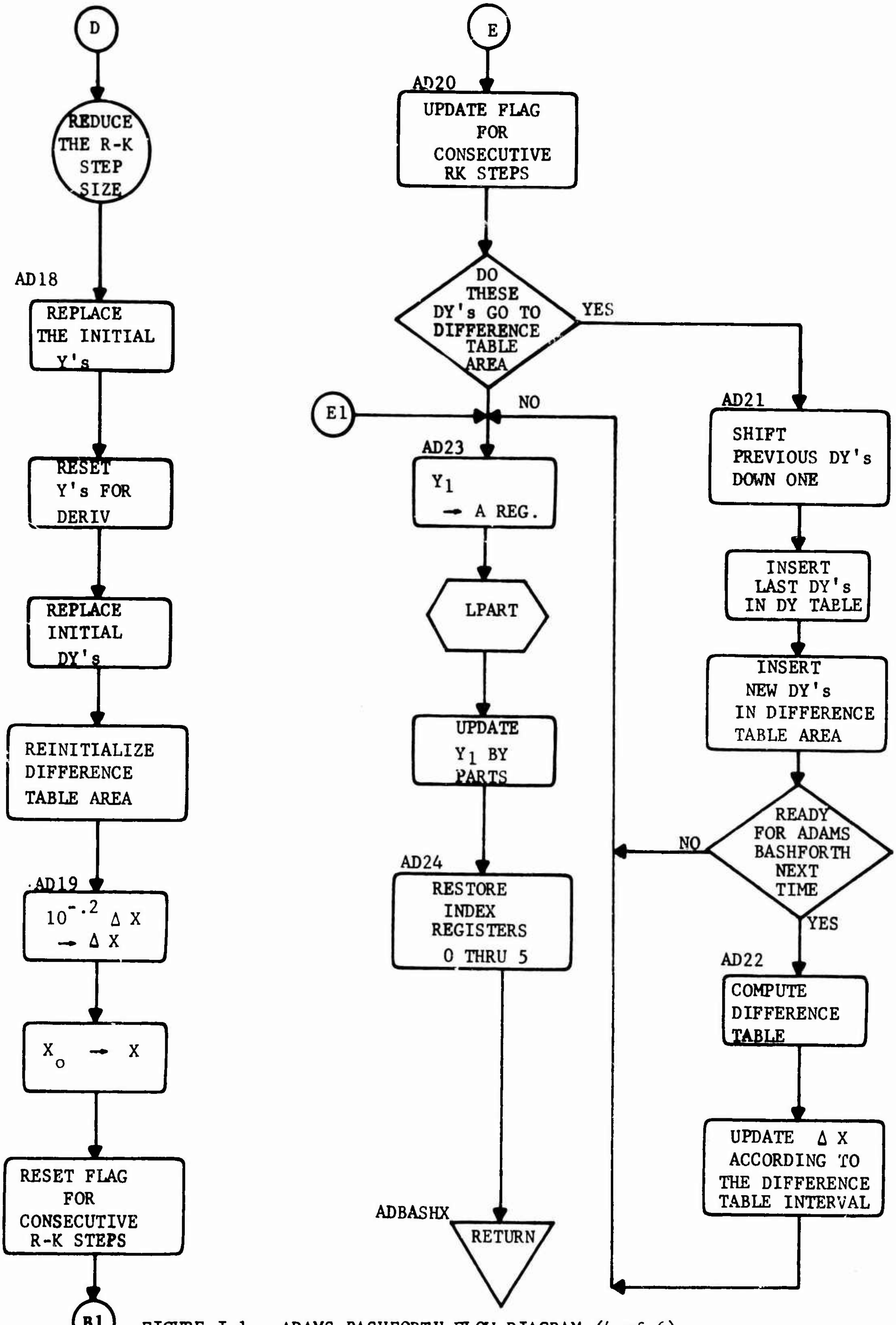


FIGURE I-1. ADAMS-BASHFORTH FLOW DIAGRAM (4 of 6)

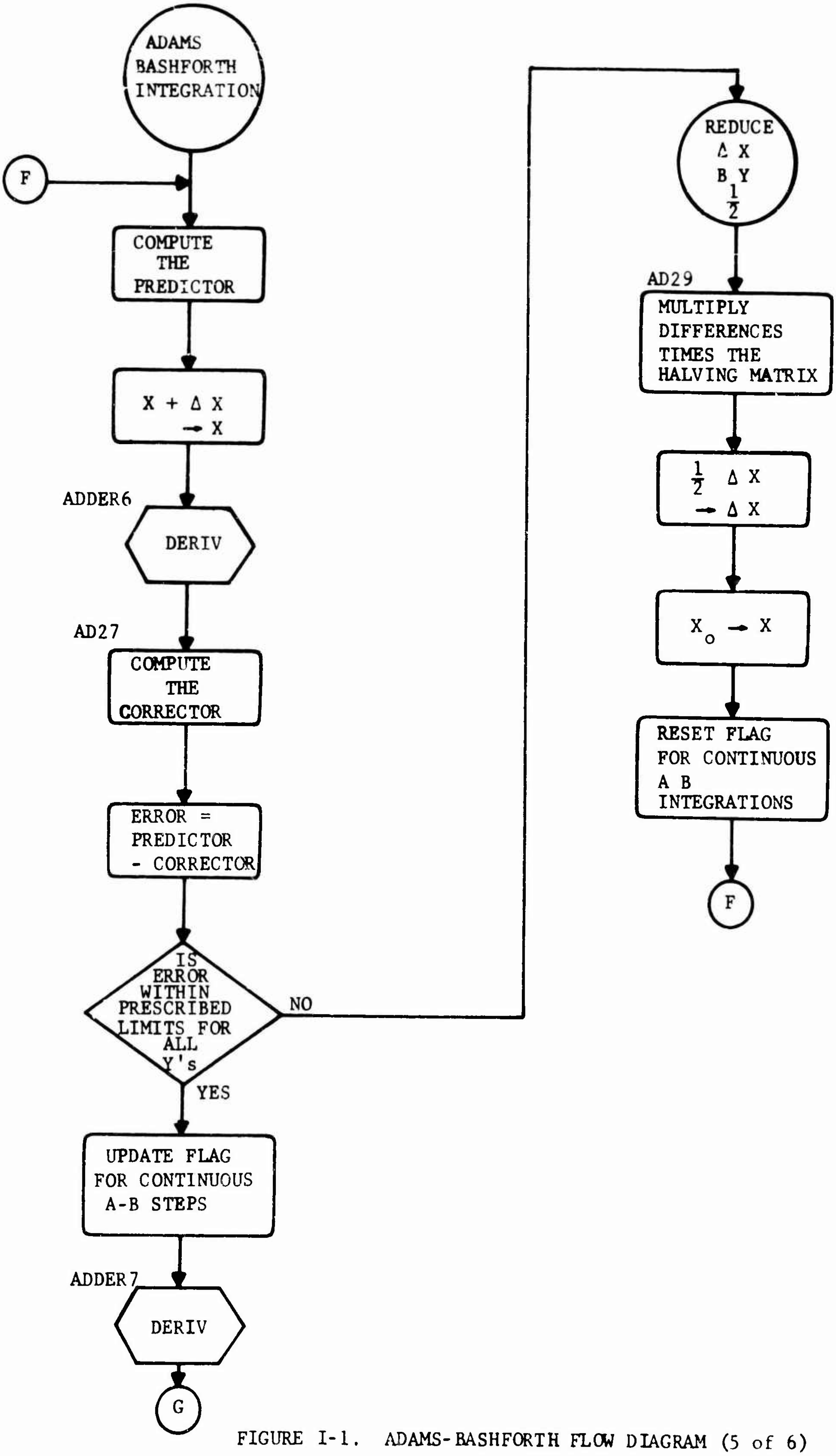


FIGURE I-1. ADAMS-BASHFORTH FLOW DIAGRAM (5 of 6)

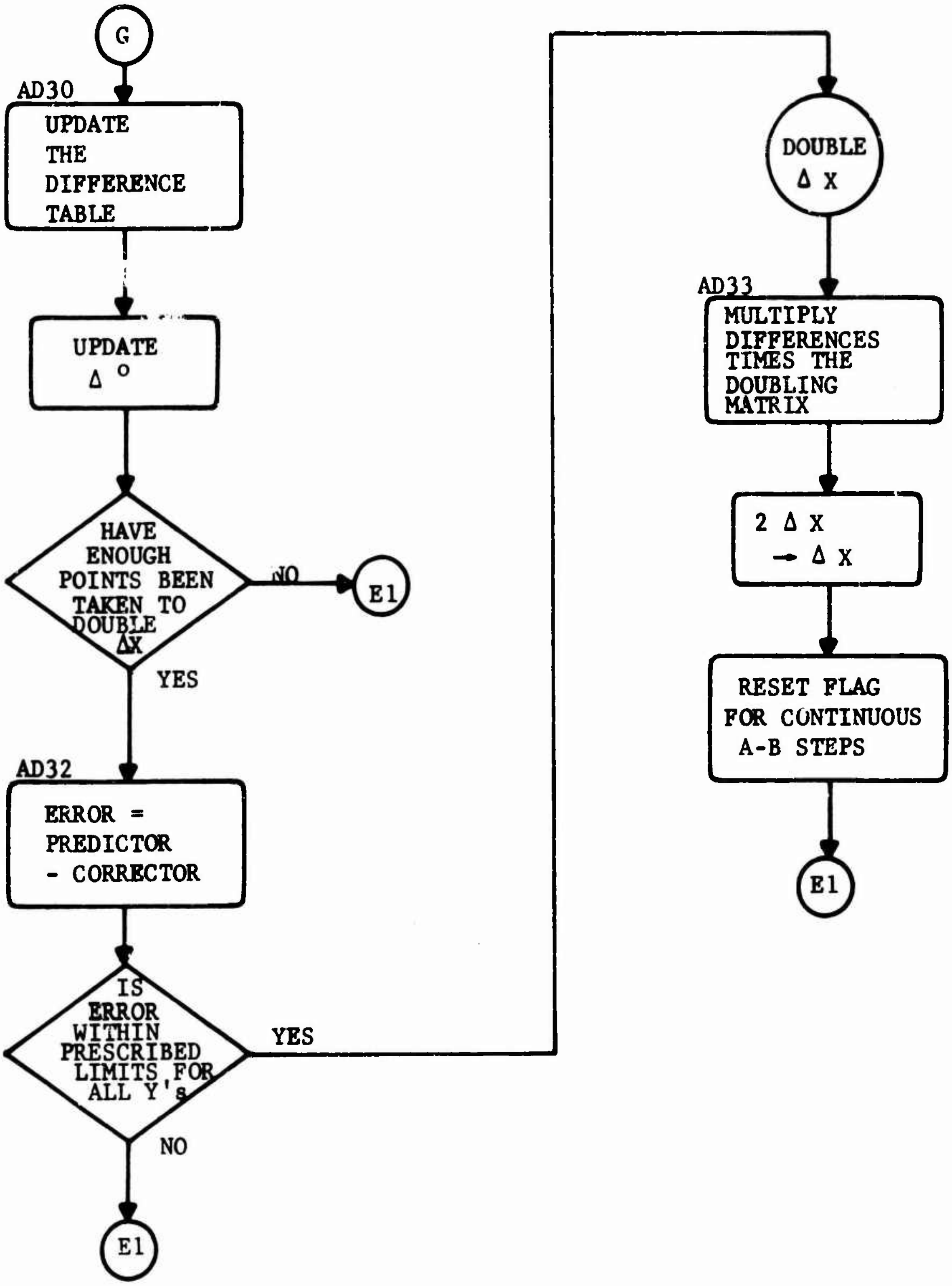


FIGURE I-1. ADAMS-BASHFORTH FLOW DIAGRAM (6 of 6)

Description of W Buffer (Work Region)

Word		
W + 0	Not Used	
+ 1	T	
+ 2	ΔT to be attempted on next entry to ADBASH	
+ 3	T at step start	
+ 4	Integration direction; + or - any non zero number	
+ 5	L	Y's
+ 6	a_x	
+ 7	a_y	
+ 8	a_z	
+ 9	h_x	
+10	h_y	
+11	h_z	
+12	ϵ_L	Absolute Error Control
+13	ϵ_{a_x}	
+14	ϵ_{a_y}	
+15	ϵ_{a_z}	
+16	ϵ_{h_x}	
+17	ϵ_{h_y}	
+18	ϵ_{h_z}	

For the variables listed above each column, the following locations contain:

L	a_x	a_y	a_z	h_x	h_y	h_z	
W + 36	W + 60	W + 84	W + 108	W + 132	W + 156	W + 180	∇_i^0
+ 37	+ 61	+ 85	+ 109	+ 133	+ 157	+ 181	∇_i^1
+ 38	+ 62	+ 86	+ 110	+ 134	+ 158	+ 182	∇_i^2
+ 39	+ 63	+ 87	+ 111	+ 135	+ 159	+ 183	∇_i^3
+ 40	+ 64	+ 88	+ 112	+ 136	+ 160	+ 184	∇_i^4
+ 41	+ 65	+ 89	+ 113	+ 137	+ 161	+ 185	∇_i^5
+ 42	+ 66	+ 90	+ 114	+ 138	+ 162	+ 186	∇_i^6

W + 187	$\dot{L}$	Derivatives
+ 188	$\dot{a}_x$	
+ 189	$\dot{a}_y$	
+ 190	$\dot{a}_z$	
+ 191	$\dot{h}_x$	
+ 192	$\dot{h}_y$	
+ 193	$\dot{h}_z$	
+ 194	L	total revolutions (radians)
+ 195	L	partial revolution (radians)
+ 196	L	Y's at step start
+ 197	a_x	
+ 198	a_y	
+ 199	a_z	
+ 200	h_x	
+ 201	h_y	
+ 202	h_z	

For the variables listed above each column, the following sets of 5 locations contain:

	L	a_x	a_y	a_z	h_x	h_y	h_z
①	W + 31	W + 55	W + 79	W + 103	W + 127	W + 151	W + 175
②	+ 32	+ 56	+ 80	+ 104	+ 128	+ 152	+ 176
③	+ 33	+ 57	+ 81	+ 105	+ 129	+ 153	+ 177
④	+ 34	+ 58	+ 82	+ 106	+ 130	+ 154	+ 178
⑤	+ 35	+ 59	+ 83	+ 107	+ 131	+ 155	+ 179

1. After a Runge Kutta step:

① $\dot{Y}$ at the half step

② Y at the step start

③ Δ Y of the second half step

④ Δ Y of the first half step

⑤ $\dot{Y}$ at the step start

2. After an Adams Bashforth step:

① Predictor

②

③

④

⑤ |Predictor - Corrector|

The predictor and corrector constants, B'^{s}, are stored as follows:

W + 26	B_o
+ 27	B_o'
+ 50	B_1
+ 51	B_1'
+ 74	B_2
+ 75	B_2'
+ 98	B_3
+ 99	B_3'
+122	B_4
+123	B_4'
+146	B_5
+147	B_5'
+170	B_6
+171	B_6'

All locations not mentioned are used for temporary working storage.

I-6.HOW TO USE ADBASH

The remainder of this Appendix gives specific instructions in the use of the program.

ADBASH has been designed to integrate a system of first order differential equations over some range of X from X_0 to X_{final}. Given a set of initial conditions, ADBASH will, on each successive jump to the subroutine, give a step by step solution to the system of equations. The subroutine will automatically control the step size so as to keep the error within the limits specified by the user.

To obtain values of Y_1, Y_2, ..., Y_7, at discrete values of X, the user should code his program to perform some type of interpolation, using points computed by ADBASH that surround the point of interest. This type of coding will allow ADBASH to work efficiently and use an optimum step size.

Any error exits from ADBASH must be coded into ADBASH by the user. An error exit is provided at α + 1H.

I 7. DERIVATIVE SUBROUTINE (SUPPLIED BY USER)

At each entry, the derivative subroutine must compute the derivatives of the system, using the current values of X and the Y_i, and store the results in the W buffer.

The first instruction of the derivative subroutine must be: TJM exit.

The return instruction of the derivative subroutine must be: exit JMP 0.

Input		Output	
W + 1	X		
+ 5	Y_1 (mod 2π)	W + 187	Y_1'
+ 6	Y_2	+ 188	Y_2'
+ 7	Y_3	+ 189	Y_3'
+ 8	Y_4	+ 190	Y_4'
+ 9	Y_5	+ 191	Y_5'
+10	Y_6	+ 192	Y_6'
+11	Y_7	+ 193	Y_7'

To minimize time, the derivative subroutine should perform only the function stated above.

8. INITIALIZATION

Before the first of any series of calls to ADBASH, the following must be defined:

<u>W (work) Region</u>

W + 1 = X_o

+ 4 = direction of integration, any + or - any non zero number

+ 5 = Y_1

+ 6 = Y_2

+ 7 = Y_3

+ 8 = Y_4

+ 9 = Y_5

+10 = Y_6

+11 = Y_7

$$\left.\begin{array}{lcl} +12 & = & \epsilon_{Y_1} \\ +13 & = & \epsilon_{Y_2} \\ +14 & = & \epsilon_{Y_3} \\ +15 & = & \epsilon_{Y_4} \\ +16 & = & \epsilon_{Y_5} \\ +17 & = & \epsilon_{Y_6} \\ +18 & = & \epsilon_{Y_7} \end{array}\right\} \begin{array}{l}\text{Absolute} \\ \text{error} \\ \text{control}\end{array}$$

<u>Control Word</u>

ADP - set this to 1/1T15

Derivative Subroutine Calling Setup

The name of the user supplied derivative subroutine must be inserted into the address portion of seven locations within ADBASH. These locations are:

ADDER1, ADDER2, ADDER3, ADDER4,
ADDER5, ADDER6, ADDER7

This may be accomplished as follows:

```
          TIJ       NAME
          TJM       ADDER1
          TJM       ADDER2
           :          :
           :          :
          TJM       ADDER7
```

Optional Initialization

RKINDT (initial step size in flt. pt.)
This is defined as 0.35 within ADBASH but may be changed by the user.

ADPERRK (number of R-K steps taken to generate one A-B point)
This is defined as 4RK/1AB within ADBASH but may be changed to any value below:

1/1 T15 = 2RK/1AB
1/1 T17 = 8RK/1AB
1/1 T18 = 16RK/1AB
1/1 T19 = 32RK/1AB

I-9. SAMPLE FLOW CHART

The following diagram illustrates a typical application using ADBASH:

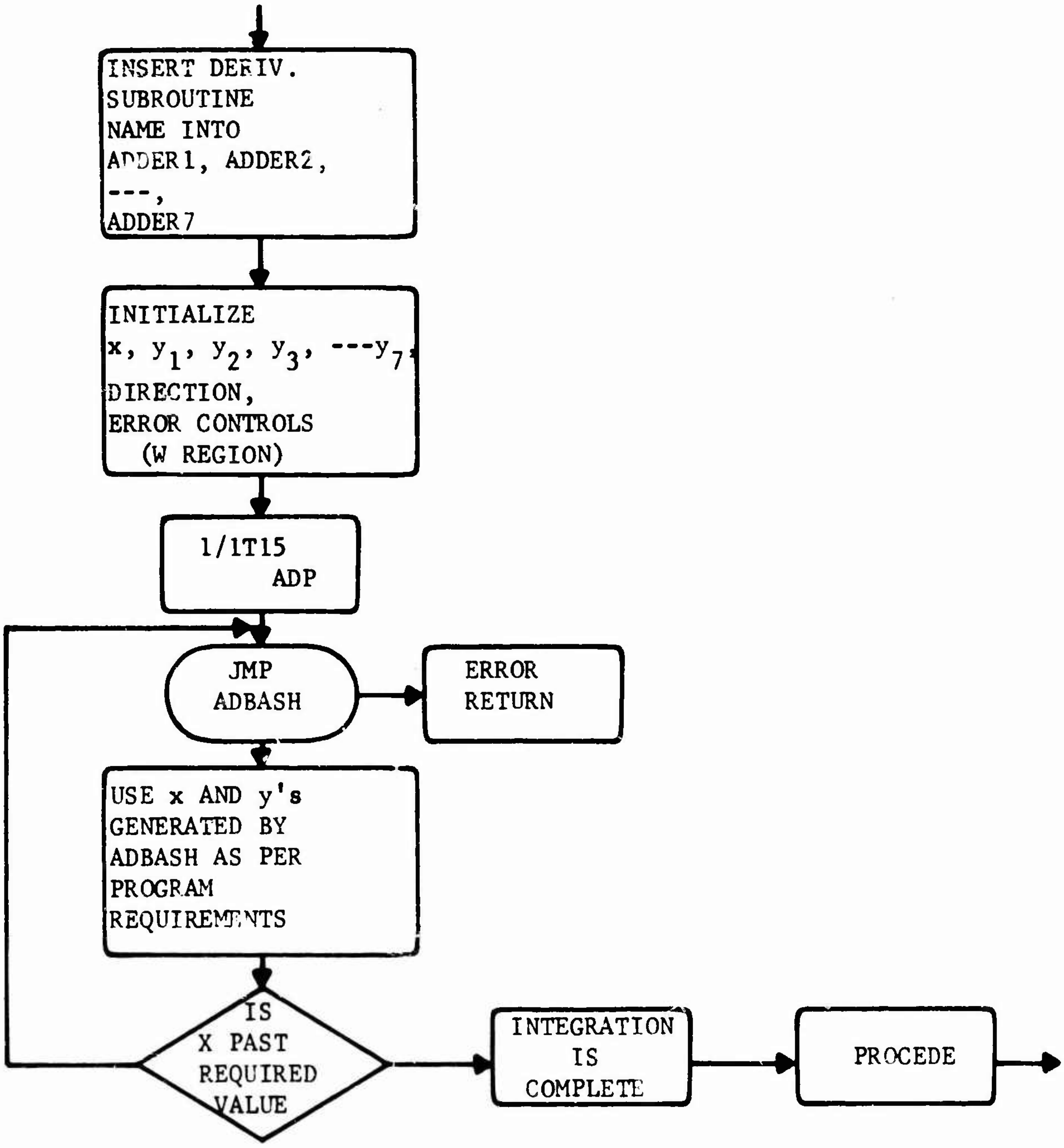


FIGURE I-2. ADBASH APPLICATION

I 10. TO USE RUNGE-KUTTA INTEGRATION ONLY

During a series of jumps to ADBASH, the user may restrict the procedure to use Runge-Kutta integration only.

To use R-K only:

1. Set ADP = 1/1T15 before the first of a series of jumps to ADBASH.
2. Before the second and each successive call to ADBASH, set ADP as follows:

   Option I: ADP = D/2.1B15; This will permit ADBASH to decrease or increase the step size as prescribed by the error controls.

   Option II: ADP = D/2.2B15; This will permit ADBASH only to decrease the step size as prescribed by the error controls.