2-76. A SUBPROGRAMS.

2-77. The subprograms in this area execute the more commonly required functions and therefore remain in core at all times with the TTP. The subprograms are as follows:

a.	ABCHEK	C33	Compare A and B parameters
b.	ARCCØS	D13	Compute Inverse Cosine
c.	ARCSIN	D12	Compute Inverse Sine
d.	ARCTAN	D14	Compute Inverse Tangent
e.	BADPT	P48	Tape or Common Error Indication
f.	BAREA	P45	Re-establish B Subprogram in Core Memory
g.	BSREC	U01	Backspace Tape One Record
h.	CØSINE	D10	Compute Cosine of Any Argument
i.	~~DLTLAT~~	~~D06~~	~~Compute Difference Between Geographic and Geocentric Latitude as a Function of Geographic Latitude~~
j.	ELLRAD	D05	Compute Earth Ellipsoid Radius as a Function of Geographic Latitude
k.	ERARSC	D51	Print Error Indication for Illegal Inverse Sine or Cosine
l.	FAD	C00	Double-Precision Addition
m.	FAPADR	U30	Binary Integer in Address to BCD
n.	FAPCMP	U30	Complement Input Into Output
o.	FAPDEC	U30	Binary Integer in Decrement to BCD
p.	FAPDGZ	U30	Binary Integer to DGZ Letter BCD

q.	FAPFIX	U30	Convert from Floating to Fixed Point
r.	FAPFLT	U30	Convert from Fixed to Floating Point
s.	FAPSTR	U30	Store Converted Number
t.	FDP	C00	Double-Precision Division
u.	FMP	C00	Double-Precision Multiplication
v.	FSB	C00	Double-Precision Subtraction
w.	GEØXYZ	D22	Convert Geocentric Coordinates to Inertial Coordinates
x.	ILLCMD	P47	Error Indication for Illegal Guidance Command
y.	INGAIN	D98	Establish Subprogram Re-entry Point
z.	INTERP	D20	Quadratic Interpolation of Single Variable
a'.	INTRØG	D50	Interrogate Simulated Sense Switch
b'.	LAGAIN	D97	Transfer to Subprogram Re-entry Point
c'.	LCTØLG	D04	Convert Geocentric Latitude to Geographic Latitude
d'.	LGTØLC	D01	Convert Geographic Latitude to Geocentric Latitude
e'.	LØCALT	D32	Determine Local Altitude Above Ellipsoid
f'.	NEGSQR	D49	Print Error Indication for Negative Square Root
g'.	RANGLE	D30	Compute Missile Re-entry Angle and Speed
h'.	RLLBCK	D39	Error Print and Rollback Control
i'.	RØUND	C32	Round Double Precision to Single Precision
j'.	RSET4	P14	Reset Index Register Four

k'.	RTRN4	D96	Return Along Saved Path
l'.	SAVE4	D95	Save Subprogram Return Path
m'.	SINE	D09	Compute Sine of Any Argument
n'.	SLIBRY	S00	Library Subroutines and Functions
o'.	SQRØØT	D15	Double-Precision Floating Point Square Root
p'.	SWAP	P46	Subprogram Time Sharing
q'.	TANGNT	D11	Compute Tangent of Any Argument
r'.	VCDØTP	D19	Compute Angle Between Two Vectors
s'.	VECMAC	D18	Compute Vector Magnitude
t'.	XPRØD	C55	Double-Precision Vector Cross Product
u'.	XYZGEØ	D27	Convert Geocentric Coordinates to Inertial Coordinates

2-78. SUBPROGRAM C33 (ABCHEK). ABCHEK checks the two channels of missile position and velocity at various points throughout the simulation to ensure agreement to within a specified amount. The FORTRAN II reference statement is CALL ABCHEK.

a. Inputs. The inputs are as follows:

COMMON TAG	DIMENSION	ITEM
LSEQ	1	Stage of flight sequence counter
LAG	1	RK step counter
FTFSP	2	Current time of flight

b. Outputs. The outputs are as follows:

COMMON TAG	DIMENSION	ITEM
FBKPS	2,2,3	Current position vector - double precision
FZETA	2,2,3	Current pitch attitude vector - double precision

The following statement is also an output:

```
(1st, 2nd) ATTEMPT FAILED AT _____ SECONDS
 A CHANNEL FOR (POSITION 3, VELOCITY 3)
```

c. Program Logic. FD C33

(1) Steps 1-5. Save the index registers. The stage of flight counter, LSEQ, and the RK counter LAG, are checked to determine if a channel variation check is to be made. LSEQ and LAG are examined and control is transferred as

follows:

LSEQ	LAG	Control is transferred to step
= 11	= 5	7
= 11	= 4	7
= 11	≠ 4 or 5	6
≠ 11	≠ 4	6
≠ 3,6,9, or 11	= 4	6
= 3,6, or 9	= 4	7

(2) Step 6. The index registers are restored and the subprograms return to the user subprogram.

(3) Steps 7-10. VECMAG computes the position vector magnitude. If an error has not previously occurred for the same time of flight, control is transferred to step 21. If the present vector is within allowable limits, control is transferred to step 28.

(4) Steps 11-12. Preparation is made for printing second failure statement and to return to step 19 after printing. Control is transferred to step 13.

(5) Steps 13-18. BAREA verifies that the print subprograms are in core, and the proper position or velocity error statement is printed.

(6) Steps 19-20. The internal channel registers holding the failure values are cleared. The subprogram exits to HALT for manual intervention.

(7) Steps 21-25. If the present vector is within allowable limits, control is transferred to step 31. The time of flight and the two channel registers are saved. Preparation is made for printing a first-failure statement and to return to step 26 after printing. Control is then transferred to step 13.

(8) Steps 26-27. IFLAG is set to octal 311 and the subprogram exits to RLLBCK to roll back to the previous check point.

(9) Steps 28-30. If neither channel A nor Channel B agree with its previous value, control is transferred to step 11, otherwise the work register used for saving the time of flight is cleared.

(10) Steps 31-32. If the velocity has not been checked, control is transferred to step 33. Otherwise, the two channels corresponding to FBKPS and FZETA are set equal and control is transferred to step 6.

(11) Steps 33-34. Set index register to allow for checking of the velocity vector differences. VECMAG computes the velocity vector magnitude and control is transferred to step 8.

2-79. SUBPROGRAM D13 (ARCCØS). ARCCØS computes the first or second quadrant angle Θ in radians. The FORTRAN II reference statement is CALL ARCCØS (BX, DA).

a. Inputs. The input is the duplexed argument BX which is the cosine of the derived angle Θ and $\pi/2$ in FHFPI. BX defines any pure number between - 1 and + 1 in single-precision floating point form. BX is of dimension two.

b. Outputs. The output is the duplexed argument DA which is arccosine BX, expressed in single-precision floating point form. DA is of dimension two.

c. Program Logic. FD D13. The first step in computing the angle is to obtain arcsine BX using ARCSIN. Using arcsine BX, ARCCØS computes two values of DA, first using expression (1) and then using expression (2). CUTIE is stepped by one and control is returned to the user subprogram.

d. Expressions.

$$\text{ARCCØS}\ (1) = -\left(\text{ARCSIN BX}\ (1) - \frac{\pi}{2}\right) \qquad (1)$$

$$\text{ARCCØS}\ (2) = -\frac{\pi}{2} - \text{ARCSIN BX}\ (2) \qquad (2)$$

2-80. SUBPROGRAM D12 (ARCSIN). ARCSIN computes the angle A (in radians) whose sine, X, is known. The FORTRAN II reference statement is CALL ARCSIN (X, A).

a. Inputs. The input is the duplexed argument X of the call statement, and the following common registers.

COMMON TAG	DIMENSION	ITEM
FHFPI	2	Floating point $\pi/2$
FCØFS	2,11	Matrix of arcsin coefficients

b. Outputs. The output is the duplexed argument A of the call statement expressed in single-precision floating point form.

c. Program Logic. FD D12

(1) Steps 1-9. Tests are made on the absolute value of the input argument to determine the mode of computation. For values of $|X| > 1.000000939$ ERARSC prints an error indication for illegal arcsine. The following tests determine the mode of computation.

TEST	MODE
$\lvert X\rvert = 1.0$	$\sin^{-1} X = \pi/2 \operatorname{sgn} X$
$1.0 < \lvert X\rvert \leq 1 + 9.39 \times 10^{-9}$	$\sin^{-1} X = \pi/2$
$\lvert X\rvert \leq 3.9 \times 10^{-3}$	$\sin^{-1} X = X$
$3.9 \times 10^{-3} < \lvert X\rvert \leq 4.2 \times 10^{-2}$	expression (1)
$4.2 \times 10^{-2} < \lvert X\rvert \leq 1.1 \times 10^{-1}$	expression (2)

TEST	MODE
$1.1 \times 10^{-1} < \lvert X \rvert < 1.0$	expression (3)

(2) Steps 10-11. A duplexed value of $\sin^{-1} X$ is produced by performing the above computations on the duplexed argument X. CUTIE is stepped by one and the subprogram returns to the user subprogram.

d. Expressions.

$$\sin^{-1} X = X \left[1 + \frac{X^2}{6}\right] \qquad (1)$$

$$\sin^{-1} X = X \left[1 + \frac{X^2}{6} + \frac{3\ X^4}{40}\right] \qquad (2)$$

$$\sin^{-1} X = \operatorname{sgn} X \left[\frac{\pi}{2} - \sqrt{1-Y}\ (a_0 + a_1 Y + a_2 Y^2 X \ldots + a_7 Y^7)\right] \qquad (3)$$

where $Y = \lvert X \rvert$

a_0 =	1.5707	9360	50
a_1 =	-0.2145	9880	16
a_2 =	0.0889	7898	74
a_3 =	-0.0501	7430	46
a_4 =	0.0308	9188	10
a_5 =	-0.0170	8812	56
a_6 =	0.0066	7009	01
a_7 =	-0.0012	6249	11

2-81. SUBPROGRAM D14 (ARCTAN). ARCTAN computes the first or fourth quadrant angle θ (in radians) whose tangent, X, is known. The FORTRAN II reference statement is CALL ARCTAN (X, A).

a. Inputs. The input is the duplexed argument X which defines any pure number expressed in single-precision floating point form. X is of dimension two.

b. Outputs. The output is the duplexed argument A which is arctan X, expressed in single-precision floating point form. A is of dimension two.

c. Program Logic. ARCTAN computes arctan X using BE - SYS 3 ATANF. This procedure gives an output angle with good accuracy near zero degrees.

2-82. SUBPROGRAM P48 (BADPT). BADPT prints an indication that an error occurred in Common or binary tape. The FORTRAN II reference statement is CALL BADPT.

a. Inputs. No inputs are defined.

b. Outputs. The output is the following statement printed and written: ERROR IN COMMON OR BINARY TAPE HOPELESS HALT.

c. Program Logic. The output statement is printed and written. Processing is halted until the error is corrected, at which time the subprogram exits to the user subprogram.

MOD F1

2-83. SUBPROGRAM P45 (BAREA). BAREA checks to see if the required B subprograms are in core and, if not, reads them in. The FORTRAN II reference statement is CALL BAREA.

a. Inputs. The inputs are the following simulated switches located in Common:

COMMON TAG	ITEM (switch in ØN state)
SW(18)	R/L tape card indicator
SW(33)	B subprograms are in core
SW(75)	ØTC card indicator
SW(76)	TØT card indicator
SW(77)	TAA card indicator
SW(78)	RSD card indicator
SW(79)	SIM card indicator
SW(80)	DEC card indicator
SW(128)	ØPC card indicator
SW(137)	B6 subprograms are in core
SW(153)	B3 subprograms are in core
SW(154)	B5 subprograms are in core
SW(155)	B7 subprograms are in core
SW(156)	B8 subprograms are in core
SW(157)	B4 subprograms are in core
SW(171)	M/T tape card indicator
SW(173)	Input tape card indicator
SW(174)	B9 subprograms are in core
SW(175)	B10 subprograms are in core

b. Outputs. The output is the placement in core of the B Common and the B subprograms required by the function being performed.

c. Program Logic. FD P45

(1) Steps 1-4. If SW(33) is ØN, the B subprograms are in core and control is transferred to step 6. If ØFF, U04 reads the B1 subprograms into core. If an error did not occur in U04 (SW(70) = ØFF), SW(33) is set ØN and control is transferred to step 6; otherwise the subprogram continues at the next step.

(2) Step 5. The subprogram exits to BADPT for manual intervention.

(3) Steps 6-17. Control is transferred to different steps depending on what the subprogram is performing as indicated by the state of various switches.

PROGRAM PERFORMING	SWITCH STATE	CONTROL TRANSFER TO STEP
R/L Tape Preparation	SW(18) = ØN	18
M/T Tape Preparation	SW(171) = ØN	21
Input Tape Preparation	SW(173) = ØN	33
Output Targeting Kit (ØPC) Preparation	SW(128) = ØN	24
ØTC	SW(75) = ØN	27
TØT	SW(76) = ØN	27
TAA	SW(77) = ØN	30

PROGRAM PERFORMING	SWITCH STATE	CONTROL TRANSFER TO STEP
RSD	SW(78) = ØN	36
SIM	SW(79) = ØN	39
DEC	SW(80) = ØN	21

If none of the switches in the preceding table is ØN, SW(70) is set ØN and a six is stored in ITYER. The subprogram exits to ERRPRT to print the error.

(4) Steps 18-20. If SW(174) is ØN, the R/L tape subprograms are in core and control is transferred to step 46. If ØFF, U04 reads the B9 subprograms into core. SW(174) is set ØN and control is transferred to step 45.

(5) Steps 21-23. If SW(137) is ØN, the DEC subprograms are in core and control is transferred to step 46. If ØFF, U04 reads the B6 subprograms into core. SW(137) is set ØN and control is transferred to step 45.

(6) Steps 24-26. If SW(157) is ØN, the ØPC subprograms are in core and control is transferred to step 46. If ØFF, U04 reads the B4 subprograms into core. SW(157) is set ØN and control is transferred to step 45.

(7) Steps 27-29. If SW(153) is ØN, the ØTC and TØT subprograms are in core and control is transferred to step 46. If ØFF, U04 reads the B3 subprograms into core. SW(153) is set ØN and control is transferred to step 45.

(8) Steps 30-35. If SW(154) is ØFF, the TAA subprograms are not in core. U04 reads the B5 subprograms into core. SW(154) is set ØN and control is transferred to step 45. If SW(154) is ØN, and the input tape subprograms are in core (SW(175) = ØN), control is transferred to step 46. If SW(175) is ØFF, U04 reads the B10 subprograms into core, SW(175) is set ØN and control is transferred to step 45.

(9) Steps 36-38. If SW(155) is ØN, the RSD subprograms are in core and control is transferred to step 46. If ØFF, U04 reads the B7 subprograms into core. SW(155) is set ØN and control is transferred to step 45.

(10) Steps 39-44. If SW(156) is ØFF, the SIM subprograms are not in core. U04 reads the B8 subprograms into core. SW(156) is set ØN, and control is transferred to step 45. If SW(156) is ØN and the input tape subprograms are in core (SW(175) = ØN), control is transferred to step 46. If SW(175) is ØFF, U04 reads the B10 subprograms into core. SW(175) is set ØN.

(11) Step 45. If SW(70) is ØN, an error has occurred in U04 and control is transferred to step 5. Otherwise the subprogram continues at the next step.

(12) Step 46. Tape A1 is rewound. The subprogram exits to the user subprogram.

2-84. SUBPROGRAM U01 (BSREC). BSREC backspaces a tape one record. The FORTRAN II reference statement is CALL BSREC(N). The FAP reference instruction is CALL BSREC,N.

a. Inputs. The input is the logical tape number (1-20) listed as the argument N in the decrement part of the location.

b. Outputs. No outputs are defined.

c. Program Logic. The contents of index register 4 are saved and index register 4 is replaced with the logical tape number. If the tape is on channel A, logical tape number 1-10, the channel A backspace record instruction is added to the logical tape number and the quantity is saved for backspacing. If the tape is on channel B, logical tape number 11-20, the channel B backspace record instruction is added to the logical tape number minus 10 and this quantity is saved for backspacing. The tape determined in the above steps is backspaced one record, the contents of index register 4 are restored, and the subprogram returns to the user subprogram.

2-85. SUBPROGRAM D10 (CØSINE). CØSINE computes the cosine of any angle expressed in radians. The FORTRAN II reference statement is CALL CØSINE (BA, DX).

a. Inputs. The input is the duplexed double-precision argument BA which defines any positive or negative angle in radians. If BA is actually single precision, the least significant registers of BA contain zeros. BA is of dimension four.

b. Outputs. The output is the duplexed argument DX which defines the computed cosine value, expressed in double-precision floating point form. If DX is actually single precision (for input angles larger than 0.1 radian), the least significant registers of DX contain zeros. DX is of dimension four.

c. Program Logic. FD D10

(1) Step 1. The absolute magnitude of the input angle determines the procedure for computing its cosine. If this value is less than or equal to 0.1 radian, the subprogram continues at step 2. If this value is greater than 0.1 radian, the subprogram continues at step 6.

(2) Steps 2-5. Expression (1) is evaluated by first computing sin BA by SINE and then multiplying sin BA by itself in FMP to obtain $(\sin BA)^2$. FSB subtracts $(\sin BA)^2$

from 1 to obtain $1 - (\sin BA)^2$. To complete the expression, SQRØØT computes the square root of $1 - (\sin BA)^2$. CUTIE is stepped by one and control is returned to the user subprogram.

(3) Steps 6-7. Expression (2) evaluates cos BA. FAD adds the angle BA (in radians) to $\frac{\pi}{2}$. To complete expression (2), SINE computes $\sin (BA + \frac{\pi}{2})$. CUTIE is stepped by one and control is returned to the user subprogram.

d. <u>Expressions</u>.

$$\cos (BA) = \sqrt{1 - (\sin BA)^2} \qquad (1)$$

$$\cos (BA) = \sin (BA + \frac{\pi}{2}) \qquad (2)$$

(2-99 and 2-100 deleted)

2-87. SUBPROGRAM D05 (ELLRAD). ELLRAD computes earth ellipsoid radius as a function of geocentric latitude with an error not greater than five feet. The FORTRAN II reference statement is CALL ELLRAD.

a. Inputs. The inputs are the geocentric latitude expressed as a decimal number of degrees and the radians to degrees conversion factor in FRTØD. Geocentric latitude is defined GCLAT.

b. Outputs. The output is the required earth radius expressed in feet. Radius is defined GCRAD.

c. Program Logic. GRASE is set to convert the latitude angle from degrees into radians for use in expression (1). Expression (1) is used to compute the earth ellipsoid radius GCRAD(1). GCRAD(1) is set equal to GCRAD(2). CUTIE is stepped by one and control is returned to the user subprogram.

d. Expressions.

$$\text{GCRAD}(1) = \frac{20925696}{\sqrt{1. + (\sin a)^2 \; .67384456 \times 10^{-2}}} \quad (1)$$

where

$$a = \text{geocentric latitude in radians}$$

$$\text{GCRAD}(2) = \text{GCRAD}(1) \quad (2)$$

2-88. SUBPROGRAM D51 (ERARSC). ERARSC prints an error indication for an illegal arcsine or arccosine. The FORTRAN II reference statement is CALL ERARSC (BX).

a. Inputs. The input is the argument BX which is a duplexed single-precision quantity larger in magnitude than unity.

b. Outputs. The output is the following statement printed or written: ILLEGAL ARCSIN OR ARCCOS ATTEMPTED. ARGUMENTS WERE _______ AND _____ .

c. Program Logic. BAREA verifies that the B subprograms and B Common are in core. The output statement is printed and written. The subprogram exits to RLLBCK for return to the previous check point.

2-89. SUBPROGRAM COO: (FAD), (FSB), (FMP), (FDP). COO performs double-precision arithmetic. The entry point to the requested function is derived from the SAP mnemonic code for that function. FAD indicates addition, FSB subtraction, FMP multiplication, and FDP division. The FORTRAN II reference statement is CALL Fxx (A, B, C).

a. Inputs and Outputs. Fxx is any one of the allowable subprogram entry point names. A, B, and C define any duplexed double-precision numbers. The function performed is A f B = C, where f is one of the four operations: add, subtract, multiply, or divide.

b. Program Logic. FD COO

(1) Steps 1-5. The desired subroutine function is stored and the contents of the index registers are saved. Complements of addresses A, B, and C are stored in IR1, IR2, and IR3, respectively and the Divide Check Indicator is set ØFF. The subprogram tests GØ for address of the desired subroutine.

(2) Steps 6-9. Subroutine for ADD is executed and the subprogram continues at step 10.

(3) Step 10. The contents of the index registers are restored and the subprogram returns to the user subprogram.

(4) Steps 11-14. Subroutine for SUB is executed and the subprogram continues at step 10.

(5) Steps 15-17. Subroutine for MPY is executed and the subprogram continues at step 10.

(6) Steps 18-24. Subroutine for DVP is executed and the subprogram continues at step 10.

(7) Steps 25-26. Divide Check Indicator is set ØFF and the subprogram halts.

c. Expressions.

FAD $C_1 + C_2 = A_1 + A_2 + B_1 + B_2$ (1)

FSB $C_1 + C_2 = A_1 + A_2 - B_1 - B_2$ (2)

FMP $C_1 + C_2 = A_1 B_1 + A_1 B_2 + A_2 B_1$ (3)

FDP $C_1 + C_2 = (A_1/B_1 + \text{remainder})$
$+ A_2/B_1 - A_1 B_2/B_1 B_1$ (4)

2-90. SUBPROGRAMS U30 (FAPFLT, FAPFIX, FAPSTR, FAPDGZ, FAPCMP, FAPDEC, FAPADR). FAPFLT converts a fixed point number to floating point; FAPFIX converts a floating point number to fixed point; FAPSTR sets the output equivalent to the input; FAPDGZ converts a binary integer in the decrement into the binary coded decimal form of the desired ground zero; FAPCMP complements the input into the output; FAPDEC converts a binary integer in the decrement into BCD; and FAPADR converts a binary integer in the address into BCD form. The FORTRAN II reference statements are: CALL FAPFLT (A,B), CALL FAPFIX (A,B), CALL FAPSTR (A,B), CALL FAPDGZ (A,B), CALL FAPCMP (A,B), CALL FAPDEC (A,B), and CALL FAPADR (A,B).

a. Inputs. The input is the argument A, where A is the address of the register containing the value to be converted.

b. Outputs. The output is the argument B, where B is the address of the register containing the converted value.

c. Program Logic.

(1) FAPFLT. The contents of index register four are saved and the fixed point number is picked up and masked with 233000000000_8 to set up the characteristic. The above octal number is added to the result, normalizing the value. The final result is stored for output and the subprogram returns to the user subprogram.

(2) FAPFIX. The contents of index register four are saved and the floating point number is picked up and added to

the above octal number, unnormalizing the result. The exponent bits are removed and the sign of the floating point number is tested. If minus, the resulting value is set minus. The final result is stored as the output value in fixed form and the subprogram returns to the user subprogram.

(3) FAPSTR. The contents of index register four are saved and the output is set equal to the input. The subprogram returns to the user program.

(4) FAPDGZ. The contents of index register four are saved. The input argument is moved to the address and converted to binary coded decimal form of the desired ground zero. The converted value is stored as the output value and the subprogram returns to the user subprogram.

(5) FAPCMP. The contents of index register four are saved. The magnitude of the input argument is complemented and stored as the output value, and the subprogram returns to the user subprogram.

(6) FAPDEC. The contents of index register four are saved. The input argument is moved to the address and converted from binary to binary coded decimal (BCD). The converted value is stored as the output argument and the subprogram returns to the user subprogram.

(7) FAPADR. The contents of index register four are saved, and the input argument is converted from binary to BCD. The subprogram returns to the user subprogram.

2-91. SUBPROGRAM D22 (GEØXYZ). GEØXYZ converts target and launch pad locations from geocentric latitude, longitude, and altitude form into the inertial earth-centered rectangular coordinate system. The FORTRAN II reference statement is CALL GEØXYZ.

a. Inputs. The inputs are as follows:

COMMON TAG	DIMENSION	ITEM	SYMBOL	UNITS
GPTLT	2	Geocentric latitude of point	L'_{CP}	degrees
GPTLN	2	Longitude of point west of Greenwich	λ_p	degrees
GPTGS	2	Geoidal separation from ellipsoid at point	h_{gsp}	feet
GPTHT	2	Altitude of point above geoid	h_p	feet
PRLØN	2	Geographic longitude of "this" radar	λ'_R	degrees
FTFSP	2	Time of flight since missile launch	t_L	seconds
GØMGA	2	Rotation rate of earth	Ω	rad/sec
PLDRE	2	Radius of earth r at L'_{CP}	r_{egp}	

b. Outputs. The output is the position vector corresponding to the point defined by the input parameters, expressed in the inertial earth-centered rectangular coordinate system. This position vector is duplexed and in double-precision form in a group of registers defined by FBKPS, and in single-precision form defined by FSPPS. FBKPS is of dimension twelve and FSPPS is of dimension six.

c. Program Logic. FD D22

(1) Steps 1-5. Expression (1) is evaluated using the set of duplexed inputs to obtain Ø used in expressions (3) and (4). Dual computations are performed to obtain the radius vector of the point from the center of the earth, using expression (2). The six core slots used to contain the least significant portion of the duplexed double-precision X, Y, and Z output coordinates are cleared to zero.

(2) Steps 6-9. The SINE and CØSINE compute the duplexed sine and cosine of Ø. The geocentric latitude of the point is converted to radians and the sine and cosine of this value are computed by SINE and CØSINE. The duplexed double-precision X, Y, and Z coordinates of vector position are evaluated by expressions (3), (4), and (5) using the set of duplexed inputs. The single-precision quantities are set equal to the most significant portion of the corresponding double-precision quantities. CUTIE is stepped by one and control is returned to the user subprogram.

d. Expressions.

$$\emptyset = \lambda_R' - \lambda_P - \Omega t_L \qquad (1)$$

$$r_{egp} = h_p + h_{gsp} + R \qquad (2)$$

$$\vec{X}_m = r_{egp} \cos L_{gp}' \cos \emptyset \qquad (3)$$

$$\vec{Y}_m = r_{egp} \cos L_{gp}' \sin \emptyset \qquad (4)$$

$$\vec{Z}_m = r_{egp} \sin L_{gp}' \qquad (5)$$

The above terms are defined in the Inputs paragraph.

2-92. SUBPROGRAM P47 (ILLCMD). ILLCMD prints a notification of an illegal guidance command situation. The FORTRAN II reference statement is CALL ILLCMD.

a. Inputs. The input is from GGDSIM and is NFLAG(10) which is the P counter.

b. Outputs. The outputs are the following statements printed and written:

a. GGDSIM ABORT

b. SUSTAINER BURNING TIME WAS TOO SHORT

c. ILLEGEL COMMANDS WERE RECEIVED FROM GGDSIM

c. Program Logic. FD P47

(1) Steps 1-5. BAREA re-establishes the B subprograms and B Common in core. The P counter is examined. If P is 40, statement a is printed and written. The subprogram continues at step 7. If P is not 40, SW(151) is examined to determine if a constraint was exceeded in the last target simulation. If ØN, statement b is printed and written; if ØFF, statement c is printed and written. In either case, the subprogram continues at step 7.

(2) IFLAG is set to identification integer 1647. The subprogram exits to RLLBCK to return to the previous check point.

2-93. SUBPROGRAMS D97, D98 and P14 (LAGAIN, INGAIN, and RSET4). LAGAIN transfers to the proper re-entry point address, INGAIN establishes the proper re-entry point used by LAGAIN, and RSET4 provides a new re-entry point and then transfers to it. The FORTRAN II reference statements are CALL LAGAIN, CALL INGAIN(X), CALL RSET4.

a. Inputs. The inputs are as follows: the program re-entry point address at the last successfully passed checkpoint in AGAIN used by LAGAIN and RSET4, the checkpoint increment for establishing a new re-entry point address in CDI4 used by RSET4, and the argument X of the INGAIN call statement. X is either 1 or 2 depending on the desired re-entry point that is to be established.

b. Outputs. The output of LAGAIN is the control re-established at the previous checkpoint. The output of INGAIN is the re-entry point stored in AGAIN. The output of RSET4 is the control re-established at some new checkpoint.

c. Program Logic.

(1) Subprogram LAGAIN. Transfer is made to the re-entry point specified by AGAIN.

(2) Subprogram INGAIN. The return path of the user subprogram is established in accordance with the argument which is an increment determined by the hierarchy of the subprogram calling INGAIN. The subprogram exits to the

user subprogram.

(3) Subprogram RSET4. A new re-entry point is established by adding the increment specified by CDI4 to the re-entry point specified by AGAIN, and transfer is made to this new re-entry point.

2-94. SUBPROGRAM D20 (INTERP). INTERP evaluates X as a function of Y by linear interpolation from a table of X's and Y's. The FORTRAN II reference statement is CALL INTERP (BU, DZ, BTABL).

a. Inputs. The simplexed single-precision input X is defined as the argument BU. BTABL defines the table of X's and Y's. The table is a two-dimensional array with entries in floating point form.

b. Outputs. The simplexed single-precision output Y is defined as the argument DZ. Y is the function of X as obtained by linear interpolation. The following printed and written statement is also an output: ILLEGAL TABLE SIZE

c. Program Logic. FD D20

(1) Steps 1-4. The first register of the input table contains the number of registers in the table. This register is interrogated to determine if enough values are available to operate INTERP successfully. If less than two values are contained in the table or the designated table length is negative, the B subprograms and B Common are verified to be in core by BAREA. The output statement is printed and written. The subprogram exits to RLLBCK for return to the previous check point. If the table contains two or more values, the subprogram continues at step 5.

(2) Steps 5-10. The Y values are obtained using simplexed inputs. Two X values, X_2 and X_1, are extracted from