UNIVERSITY OF CALIFORNIA, SAN DIEGO
SCRIPPS INSTITUTION OF OCEANOGRAPHY
VISIBILITY LABORATORY
LA JOLLA, CALIFORNIA 92093

THE CZCS GEOLOCATION ALGORITHMS

Wayne H. Wilson
Raymond C. Smith
Jeffrey W. Nolten

SIO Ref. 81-32
October 1981

Supported by

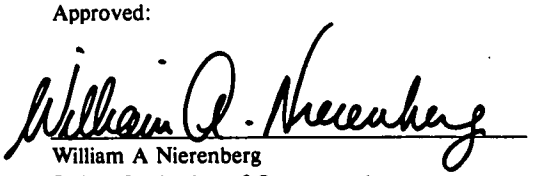
National Aeronautics and Space Administration
Contract No. NAS 5-26249

US Department of Commerce
National Oceanic and Atmospheric Administration
Grant No. NA80AA-D-00007

Approved:

Roswell W. Austin, Director
Visibility Laboratory

Approved:



William A Nierenberg
Scripps Institution of Oceanography

PREFACE

The Coastal Zone Color Scanner (CZCS) on board the Nimbus 7 satellite was designed to measure surface radiance upwelled from the ocean in 6 spectral bands. The CZCS spectrometer obtains its information from a rotating mirror and is timed to collect data when the mirror views the earth surface between *ca.* 40° to the left and right of the subsatellite track. Each scan is divided into 1968 picture elements, pixels, of 0.04 degrees scan each. In order to avoid direct reflected sun glint, the rotating mirror shaft can be tilted so that scans cross the subsatellite track up to 20° forward or aft of the point directly beneath the satellite. The CZCS is the first satellite borne instrument to have this tilted scan capability and therefore poses some new problems in locating the earth surface position of viewed pixels. This report restates results of analyses performed by several members of the Nimbus Experiment Team for the CZCS, using a consistent coordinate system and terminology.

This work was supported by the National Aeronautics and Space Administration, Contract No. NAS 5-26249, and the US Department of Commerce, National Oceanic and Atmospheric Administration, Grant No. NA80AA-D-00007.

TABLE OF CONTENTS

THE CZCS GEOLOCATION ALGORITHMS

The location on earth of a pixel viewed at a certain tilt and scan angle is a complex problem in CZCS viewing geometry and spherical trigonometry. Presented here is the derivation of the Visibility Laboratory algorithm for CZCS pixel location. It was originally developed by Wayne Wilson and Raymond Smith of the Visibility Laboratory, in collaboration with Jim Mueller of the Naval Post Graduate School, Monterey and Howard Gordon of the University of Miami.

SATELLITE GEOMETRY

Fig. 1 defines the coordinate system used relative to the satellite. *Note:* angles are symbolized by greek letters, vectors are symbolized by lower case letters superscripted with a bar ($\bar{a}$), and rotation operators, (matricies) are symbolized with capital letters.

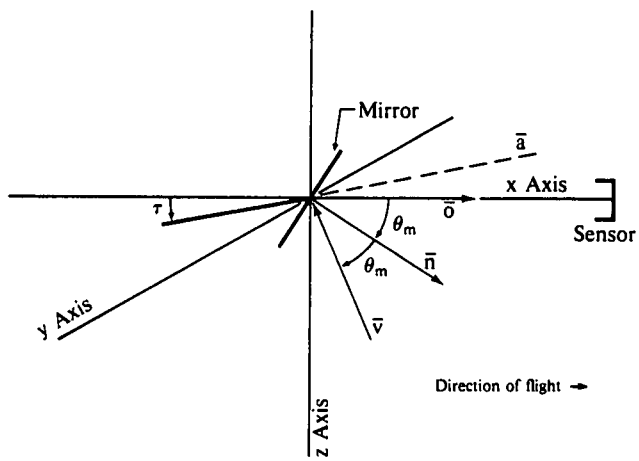


Fig. 1

DEFINITIONS

x axis	The x axis is along the direction of flight of the satellite, positive towards the radiometer sensor.
z axis	The z axis connects the center of the scan mirror and the center of the earth, positive towards the earth and orthogonal to x axis at the point of intersection.
y axis	The y axis is orthogonal to the x and z axes.
$\bar{a}$	$\bar{a}$ is the vector through the shaft of the scan mirror (the mirror rotates about this vector). Vector $\bar{a}$ rotates about the y axis in the x-z plane defining the tilt angle τ.
$\bar{n}$	$\bar{n}$ is a unit vector orthogonal to the surface of the scan mirror.
$\bar{o}$	$\bar{o}$ is a unit vector along the x axis (oriented toward the sensor).
$\bar{v}$	$\bar{v}$ is a unit vector from the direction of the pixel to the scan mirror.
	Note that vectors $\bar{o}$, $\bar{n}$, and $\bar{v}$ *lie in the same plane.*
θ_m	θ_m is the angle between $\bar{o}$ and $\bar{n}$.
τ	τ is the tilt angle of the scan mirror shaft. (One half the "tilt" given in the subcommutated data.)
η	η (not shown) is the mirror shaft rotation angle.

The tilt and scan angle are known parameters. From these we would like to calculate the nadir angle θ from the z axis to vector $\bar{v}$, and the azimuth angle ϕ from the x axis to the projection of $\bar{v}$ on the x-y plane (Fig. 2).

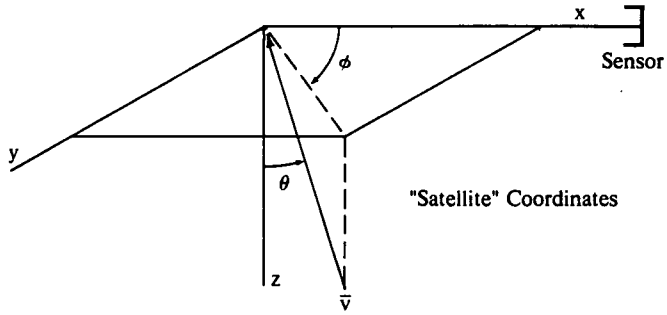


Fig. 2

For convenience in the derivations to follow it is desirable to rotate the coordinate system above to a "natural" coordinate system such that $z_n = x$, $x_n = -z$, and $y_n = y$ (Fig. 3). In this system the angles describing $\bar{v}$, (θ_n and ϕ_n) can be more easily calculated from τ and η. After $\bar{v}$ is described in terms of θ_n and ϕ_n the coordinates will be rotated to the "satellite" system and θ and ϕ derived.

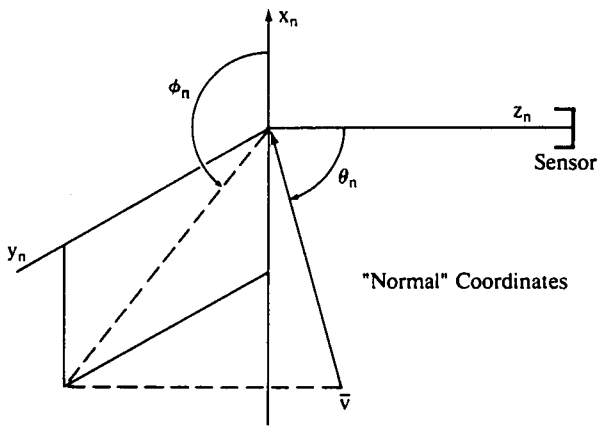


Fig. 3

Recall that vector $\bar{n}$ is the normal to the scan mirror and therefore is dependant only on tilt and scan for its orientation. The initial position of vector $\bar{n}$ at zero tilt and zero scan can be defined as lying in the x_n–z_n plane at an angle 45° clockwise rotation from the z_n axis, (Fig. 4).

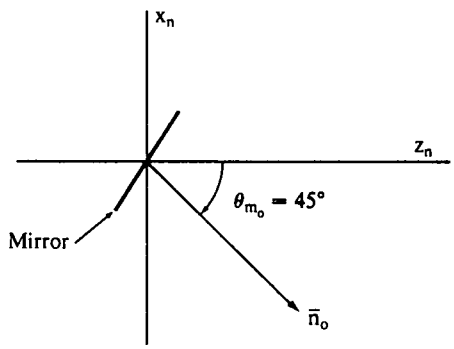


Fig. 4

Since $\bar{n}$ is a unit vector, its x_n, y_n, z_n coordinates may be calculated:

$$\bar{n}_0 = \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} = \begin{bmatrix} \cos\theta_m \\ 0 \\ \sin\theta_m \end{bmatrix} = \begin{bmatrix} -1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{bmatrix} = \begin{bmatrix} -a \\ 0 \\ a \end{bmatrix} .$$

The CZCS mirror tilts about the y_n axis, (assumed to be through the surface of the mirror). The tilt angle τ is defined positive when $\bar{n}$ rotates in a counter clockwise direction. An Eulerian rotational operator T can be defined such that it acts on $\bar{n}$ to cause a counter clockwise rotation about y_n.

$$T = \begin{bmatrix} \cos\tau & 0 & \sin\tau \\ 0 & 1 & 0 \\ -\sin\tau & 0 & \cos\tau \end{bmatrix} .$$

The mirror scans about the mirror shaft; at zero tilt this shaft lies along the z_n axis. An Eulerian rotational operator S can be described to rotate $\bar{n}$ around the z_n axis. When facing the negative z_n axis direction, this rotation is positive in a clockwise direction.

$$S = \begin{bmatrix} \cos\eta & \sin\eta & 0 \\ -\sin\eta & \cos\eta & 0 \\ 0 & 0 & 1 \end{bmatrix} .$$

We would like to combine the operators into a single operator. Normally one considers the CZCS mirror to scan about a tilted mirror shaft, however if we operate T on $\bar{n}$ the mirror shaft (fixed in relation to $\bar{n}$) also rotates out of alignment with z_n and the use of S to scan about z_n becomes invalid. It is mathematically valid to scan $\bar{n}$ about z_n first then tilt it about y_n second. Vector $\bar{n}$ will have the same final coordinates and the simpler Eulerian operators can be used.

$$TS = T(S) = \begin{bmatrix} \cos\tau\cos\eta & \cos\tau\sin\eta & \sin\tau \\ -\sin\eta & \cos\eta & 0 \\ -\sin\tau\cos\eta & -\sin\tau\sin\eta & \cos\tau \end{bmatrix} .$$

Therefore for any τ and η,

$$\bar{n}' = \begin{bmatrix} x_n' \\ y_n' \\ z_n' \end{bmatrix} = TS \begin{bmatrix} -a \\ 0 \\ a \end{bmatrix} = \begin{bmatrix} a(\sin\tau - \cos\tau\cos\eta) \\ a\sin\eta \\ a(\cos\tau + \sin\tau\cos\eta) \end{bmatrix} .$$

We now have the coordinates for the mirror normal vector $\bar{n}$ after scan and tilt and can proceed to derive the coordinates of vector $\bar{v}$ from the direction of the pixel. Recall that vector $\bar{o}$ is a unit vector lying along the z_n axis in the direction of the CZCS sensor.

$$\bar{o} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} .$$

Vector $\bar{o}$ is separated from vector $\bar{n}$ by the angle θ_m. Because the angle of incidence to a mirror is equal to the angle of reflection, vector $\bar{v}$ is also separated from $\bar{n}$ by angle θ_m and lies in the same plane as vectors $\bar{o}$ and $\bar{n}$.

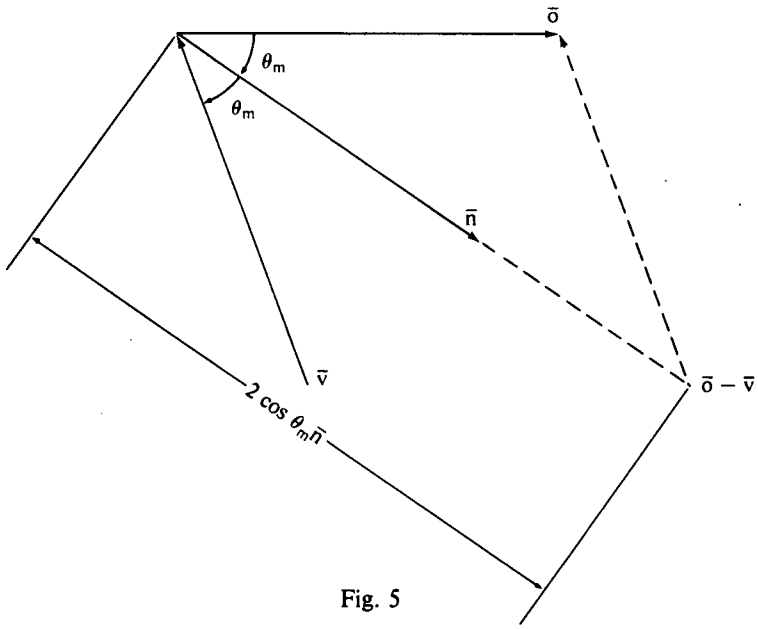


Fig. 5

Since vector $\bar{v}$ is directed into the mirror and vector $\bar{o}$ is directed away from the mirror, if vector $\bar{v}$ is subtracted from vector $\bar{o}$ the resulting vector will lie along $\bar{n}$ with a length of $2 \cos \theta_m$, (Fig. 5).

$$\bar{o} - \bar{v} = \bar{n}\, 2\cos \theta_m$$

$$\bar{v} = \bar{o} - \bar{n}\, 2\cos \theta_m$$

$$\bar{v} = \begin{bmatrix} 0 - 2\cos \theta_m \cdot a \cdot (\sin \tau - \cos \tau \cos \eta) \\ 0 - 2\cos \theta_m \cdot a \cdot \sin \eta \\ 1 - 2\cos \theta_m \cdot a \cdot (\cos \tau + \sin \tau \cos \eta) \end{bmatrix} .$$

With the coordinates of $\bar{v}$, angle ϕ_n can now be calculated,

$$\tan \phi_n = \frac{y_n}{x_n} = \frac{\sin \eta}{\sin \tau - \cos \tau \cos \eta} .$$

Note that computational machines return arctangent functions in the range of $\pm 90°$. Therefore for normal scan angles $180°$ must be added to the returned value of ϕ_n, (Fig. 6). Therefore:

$$\phi_n = 180° + \arctan(\sin \eta \,/\, (\sin \tau - \cos \tau \cos \eta)) .$$

Since $|\bar{o}| = |\bar{n}| = 1$,

$$\bar{n} \cdot \bar{o} = |\bar{o}|\, |\bar{n}| \cos \theta_m = \cos \theta_m = a(\cos \tau + \sin \tau \cos \eta) ,$$

$$\theta_n = 2\theta_m ,$$

$$\theta_n = 2 \arccos\left(a(\cos\tau + \sin\tau \cos\eta)\right) \quad .$$

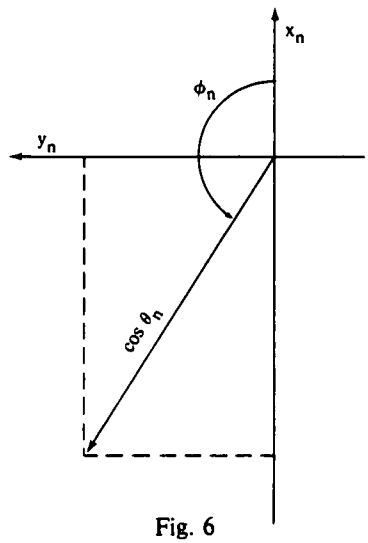


Fig. 6

Vector $\bar{v}$ is now described in the "normal" coordinate system in terms of the angles θ_n and ϕ_n. To rotate back to the "satellite" coordinate system recall that:

$$\begin{matrix} & \text{"satellite"} & & \text{"normal"} \end{matrix}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{bmatrix} = \begin{bmatrix} z_n \\ y_n \\ -x_n \end{bmatrix} = \begin{bmatrix} \cos\theta_n \\ \sin\theta_n \sin\phi_n \\ -\sin\theta_n \cos\phi_n \end{bmatrix} \quad . \tag{1}$$

Therefore:

$$\theta = \arccos(-\sin\theta_n \cos\phi_n)$$

and

$$\phi = \arctan(y/x) = \arctan(\tan\theta_n \sin\phi_n) \; .$$

Due to the quadrant ambiguity of the arctangent function, for negative tilts, 180° may need to be added to ϕ to correct ϕ.

PITCH - ROLL - YAW

The "satellite" coordinate system as defined above is fixed relative to the space craft. However, the satellite may be rotated due to pitch, roll, and/or yaw relative to an "earth" coordinate system. The origin of the "earth" system is the same as for the "satellite" system, the center of the satellite. The x direction is the instantaneous tangent to the flight path, the z direction is towards the center of the earth, and y is orthogonal to x and z. The "satellite" system will oscilate about the "earth" system by small angles generally less than a degree. Fig. 7 shows the directions of the pitch, roll, and yaw rotations on the "earth" system, these rotations define the orientation of the "satellite" system in Fig. 2.

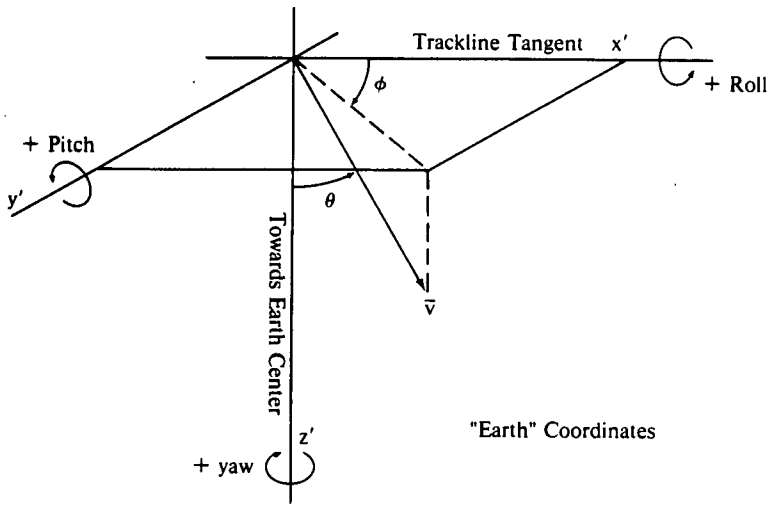


Fig.7.

At this point we have vector $\bar{v}$ defined by angles ϕ and θ relative to the "satellite" coordinate system. The x, y, and z coordinates of $\bar{v}$ may be calculated as in Eq.(1) above:

$$x = \sin\theta \cos\phi$$
$$y = \sin\theta \sin\phi$$
$$z = \cos\theta \quad .$$

We now need to create a set of operators that will rotate the "satellite" x, y, z coordinates to a set of "earth" x', y', z' coordinates for $\bar{v}$. The order of correction is significant. The xyz coordinates must first be pitched about the y axis, then rolled about the new x axis and finally yawed about the again new z axis. First the pitch operator:

$$P = \begin{pmatrix} \cos p & 0 & \sin p \\ 0 & 1 & 0 \\ -\sin p & 0 & \cos p \end{pmatrix} ,$$

where p equals the pitch angle. Next the roll operator:

$$R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos r & -\sin r \\ 0 & \sin r & \cos r \end{pmatrix} .$$

Finally the yaw operator:

$$Y = \begin{pmatrix} \cos y & -\sin y & 0 \\ \sin y & \cos y & 0 \\ 0 & 0 & 1 \end{pmatrix} .$$

These may be combined by first multiplying R on P and then multiplying Y on RP yielding YRP.

$$YRP = \begin{pmatrix} \cos y \cos p + \sin y \sin r \sin p & -\sin y \cos p + \cos y \sin r \sin p & \cos r \sin p \\ \sin y \cos r & \cos y \cos r & -\sin r \\ -\cos y \sin p + \sin y \sin r \cos p & -\sin y \sin p + \cos y \sin r \cos p & \cos r \cos p \end{pmatrix} .$$

Since we are dealing with small angles, considerable computation time can be saved by assuming the cosine of r, p, or y = 1, (cos 1° = 0.9998). If working in the radians mode one can assume that the sine of r, p, or y equals r, p, or y. Therefore, YRP becomes:

$$YRP = \begin{pmatrix} 1 + yrp & -y + rp & p \\ p & 1 & -r \\ -p + yr & -yp + r & 1 \end{pmatrix} .$$

If working in the degrees mode, first precompute sin r, sin p, and sin y.

Now we may multiply YRP on a set of "satellite" coordinates and calculate a set of "earth" coordinates for $\bar{v}$. For clarity let the satellite x, y, z coordinates be expressed as u, v, w to distinguish them from the angles r, p, and y. Then:

$$\begin{aligned} x' &= (1 + yrp)u + (-y + rp)v + pw \\ y' &= yu + v + rw \\ z' &= (-p + yr)u + (-yp + r)v + w \end{aligned} .$$

Notice that if roll, pitch, and yaw are zero, x' = u, y' = v, and z' = w as we would expect.

Finally, we can calculate θ' and ϕ' for the "earth" system:

$$\theta' = \arccos(z')$$

$$\phi' = \arctan(y'/x') .$$

Due to the quadrant ambiguity of the arctangent function, for negative tilts, 180° may need to be added to ϕ' to correct ϕ'.

EARTH LOCATION

If the latitude and longitude of a geographical position, and the distance and true bearing to a second geographical position are known, the latitude and longitude of the second geographical position can be calculated. Distance and true bearing from the satellite ground point to the pixel can be derived from θ and ϕ, or θ' and ϕ' if correcting for roll, pitch, and yaw.

Given the orbital altitude of the satellite, h, for example 952km, and the nadir angle θ, the surface distance, δ, in degrees great circle arc is:

$$\delta = \arcsin(R \sin\theta) - \theta$$

where

$$R = (r + h)/r = 1.1494,$$

and

$$r = \text{earth mean radius} = 6371.2\text{km} .$$

The inclination of the CZCS orbital trackline is a function of latitude, and is 9.28° (west of north) at the equator. For any latitude, the inclination, α, is:

$$\alpha = \arcsin(\sin 9.28/\cos \text{lat}) .$$

True bearing, β, then is $\phi - \alpha$, if β is negative, $\beta = 360 + \beta$.

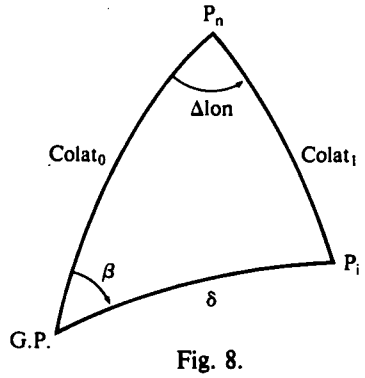


Fig. 8.

Fig. 8 represents a general spherical triangle.

Points:

G.P.	G.P. is the ground point, subsatellite point, of the satellite.
P_n	P_n is the north pole.
P_i	P_i is the pixel.

Angles:

β	β is the true bearing from the satellite G.P. to the pixel.
Δlon	Δlon is the difference in longitudes between points G.P. and P_i.
δ	δ is the great circle distance between G.P. and P_i.
$Colat_0$	$Colat_0$ is the co-latitude of the G.P. (Colat = 90° - lat)
$Colat_1$	$Colat_1$ is the co-latitude of the pixel.

If the latitude and longitude of the G.P. are known then the latitude and longitude of the pixel may be calculated directly. From the cosine rule:

$$lat_1 = \arcsin(\sin lat_0 \cos \delta + \cos lat_0 \sin \delta \cos \beta) \quad . \tag{2}$$

A lat or lon subscripted with a 0 (lat_0) refers to the G.P. and those subscripted with a 1 (lat_1) refer to the pixel. Notice that the sine of the colatitude equals the cosine of the latitude, etc. From the sine rule

$$\Delta lon = \arcsin(\sin \beta \sin \delta / \cos lat_1) \quad . \tag{3}$$

Then,

$$lon_1 = lon_0 - \Delta lon \quad .$$

These calculations work for all latitudes north (+) or south (-) to ±80.72° and all longitudes, (+ west, - east), for the north bound satellite track.

If the latitude and longitude of the pixel are known, then the calculation of the latitude of the G.P. is a complex quadratic equation since the true bearing, β, is a function of the unknown lat_0. From above:

$$\beta = \phi - \arcsin(\sin 9.28/\cos lat_0) \quad . \tag{4}$$

Let

$$I = \sin 9.28/\cos lat_0 \ .$$

Then

$$\beta = \phi - \arcsin I,$$

and we may substitute this in Eq. (2) above.

$$lat_1 = \arcsin\ (\sin lat_0 \cos \delta + \cos lat_0 \sin \delta \cos(\phi - \arcsin I)\ ) \ .$$

The term containing I may be rewritten:

$$\cos(\phi - \arcsin I) = \cos \phi \cos \arcsin I + \sin \phi \sin \arcsin I \quad . \tag{5}$$

Note that

$$\cos \arcsin I = \sqrt{1 - I^2} \quad .$$

Therefore Eq.(5) equals:

$$= \cos \phi \sqrt{1 - I^2} + I \sin \phi$$

$$= \cos \phi \sqrt{1 - \sin^2 9.28 \ / \cos^2 lat_0} + \sin \phi \sin 9.28/\cos lat_0$$

$$= \frac{\cos \phi}{\cos lat_0} \sqrt{\cos^2 lat_0 - \sin^2 9.28} + \frac{\sin \phi \sin 9.28}{\cos lat_0} \quad .$$

Substituting this last back into Eq. (2) the lat_0 terms cancel yielding:

$$\sin lat_1 = \sin lat_0 \cos \delta + \sin \delta\ (\cos \phi \sqrt{\cos^2 lat_0 - \sin^2 9.28} + \sin \phi \sin 9.28) \quad .$$

This can be rearranged yielding:

$$\sin lat_0 \cos \delta + \sin \delta \cos \phi \sqrt{1 - \sin^2 lat_0 - \sin^2 9.28} = \sin lat_1 - \sin \delta \sin \phi \sin 9.28 \quad . \tag{6}$$

Let

$$x = \sin lat_0$$
$$a = \sin lat_1 - \sin \delta \sin \phi \sin 9.28$$
$$b = \cos \delta$$
$$c = \sin \delta \cos \phi$$
$$d = 1 - \sin^2 9.28 = \cos^2 9.28 \quad .$$

Equation (6) becomes:

$$bx + c\sqrt{d - x^2} = a \quad .$$

Rearranging and removing the radical yields:

$$x^2(b^2 + c^2) - x(2ab) + (a^2 - c^2 d) = 0 \quad .$$

Thus:

$$x = \frac{ab \pm \sqrt{a^2 b^2 - (b^2 + c^2)(a^2 - c^2 d)}}{b^2 + c^2} \quad .$$

The value under the radical is chosen to be positive if tilt, (τ), is negative and negative if tilt is positive. Then:

$$lat_0 = \arcsin x \quad .$$

Note: It is simplest to precompute the values of a, b, c, and d before attempting to calculate x.

Beta is now calculated as in Eq. (4), and Δlon is calculated as in Eq. (3). Lon_0 then is:

$$lon_0 = lon_1 + \Delta lon \quad .$$

TABLE OF USEFUL INFORMATION

Mean satellite orbital period (Nov-Dec 1980) = 104.07 minutes
Mean orbital velocity (great circle arc) = 5.765×10^{-2} degrees/second
Mean satellite orbital altitude (Nov-Dec 1980) = 952 kilometers
Satellite scan rate = 123.75 milliseconds/scan line
Trackline inclination angle = 9.28 degrees west of north
Mean earth radius = 6371.2 kilometers
Earth rotational velocity = 4.1667×10^{-3} degrees/second

PROGRAM LISTINGS

The following are listings of two sets of programs using the CZCS geolocation algorithms. First is a HPL program for the H.P. 9825 calculator. This program is primarily a set of subroutines which may be used for special stand alone applications. The calling routines, accessed by special function keys, provide basic calculations and may be easily modified by for-next loops to provide for example, trackline latitudes and longitudes.

Next, is a set of Fortran programs used in the Visibility Laboratory's CZCS atmospheric correction algorithm. These subroutines, "CZSUB1" - "CZSUB5" and "CZSUBA", provide all the basic algorithm calculations. Also included is "CZNAV", a utility program similar to the HPL program, which calls the various CZ Subroutines. All the Fortran programs are written to compile on a Prime 550 Computer. "CZNAV" is written in Fortran 77. The CZ Subroutines have been compiled using Fortran 77 but were written using Fortran 66 protocol.

```
0: "CZNAV":
1: "Stand alone CZCS tracking program":
2: "09 January 1981, VisLab, Nolten":
3: dim L[6],M[2],C$[32]
4: getk "CZKEY"
5: 6→F;ent "Enter output device.",F;gsb "Notes"
6: gto "START"
7:
8: "----":for K=1 to 35;wtb F," ";next K;wtb F,"----------",13,10;ret
9:
10: "dm":
11: "returns angles in degrees.minutes decimal minutes":
12: prnd(p1,-6)→p1;ret int(p1)+.6frc(p1)
13: "dd":
14: "returns angles in decimal degrees":
15: ret int(p1)+frc(p1)/.6
16:
17: "dist":
18: "p1 = nadir angle; returns p2 = distance":
19: asn((952+6371)/6371)sin(p1))→p2;p2-p1→p2
20: ret p2
21:
22: "pix":
23: "p1=tilt, p2=pixel #, returns: p3=azimutn, p4=nadir ang, p5=shaft ang":
24: -39.04+.04p2→p5
25: 2acs((cos(p1)+sin(p1)cos(p5))/√2)→p6
26: 180+atn(sin(p5)/(sin(p1)-cos(p1)cos(p5)))→p7
27: "theta":acs(-sin(p6)cos(p7))→p4
28: if p6=90 or p6=270;sgn(p5)*2*180-90sgn(p5)→p3;jmp 2
29: "phi":atn(sin(p6)sin(p7)/cos(p6))→p3;if p1<0;180+p3→p3
30: if not (W or X or Y);ret
31: "Roll-Pitch-Yaw":
32: "x,y,z":sin(p4)cos(p3)→p8;sin(p4)sin(p3)→p9;cos(p4)→p10
33: "sin (R.P.Y)":sin(W)→p11;sin(X)→p12;sin(Y)→p13
34: "x'":(1+p11p12p13)p8+(-p13+p11p12)p9+p12p10→p14
35: "y'":p13p8+p9-p11p10→p15
36: "z'":(-p12+p11p13)p8+(-p12p13+p11)p9+p10→p16
37: "theta'":acs(p16)→p4
38: if p14;atn(p15/p14)→p20;if T<0;180+p20→p20
39: if not p14;sgn(p15)90→p20;if T<0;180+p20→p20
40: "phi'":p20→p3
41: ret
42:
43: "pixdir":
44: "entry: p1=track line azimutn, p2=G.P. lat; return: p3=true bearing":
45: p1-asn(cos(80.72)/cos(p2))→p3;if p3<0;360+p3→p3
46: ret p3
47:
48: "pixlat":
49: "entry: p1=G.P. lat, p2=stc dist, p3=true bearing; return: p4=pix lat":
```

```
50: 90-p1→p5;acs(cos(p5)cos(p2)+sin(p5)sin(p2)cos(p3))→p4;90-p4→p4
51: ret p4
52:
53: "pixlon":
54: "entry: p1=Pix. lat, p2=G.P. lon, p3=sfc dist, p4=true bearing":
55: "return: p5=pixel longitude":
56: p2-asn(sin(p4)sin(p3)/sin(90-p1))→p5
57: ret p5
58:
59: "equa":
60: "entry:p1=GP lat, p2=GP lon; return: p3=sec since, p4=deg since":
61: "p5=scan line since, p6=longitude of equator":
62: asn(sin(p1)/sin(80.72))→p4;p4/5.765e-2→p3
63: p3/.12375→p5;p2-asn(tan(p1)/tan(80.72))-4.1667e-3p3→p6;ret p6
64:
65: "scanli":
66: "entry: p1=seconds, p2=degrees; return: p3=scan lines":
67: if p2≠0;p2/5.765e-2→p1
68: p1/.12375→p3;ret p3
69:
70: "second":
71: "entry: p1=scan lines, return: p2=seconds, p3=degrees":
72: .12375p1→p2;p2*5.765e-2→p3;ret p2
73:
74: "latlon":
75: "entry: p1=new sec since equa, p2=lon of equa":
76: "return: p3=new lat, p4=new lon":
77: 5.765e-2p1→p5;asn(sin(80.72)sin(p5))→p6
78: p2+asn(tan(p6)/tan(80.72))+4.1667e-3p1→p4;p6→p3;ret
79:
80: "GP":
81: "p1=Azimuth, p2=Distance, p3=Lat-1; ret p4=Lat-0":
82: sin(p3)-sin(p2)sin(p1)sin(9.28)→r1
83: cos(p2)→r2
84: sin(p2)cos(p1)→r3
85: cos(9.28)^2→r4
86: r1r2→r5;r2^2+r3^2→r6
87: r5^2-r6(r1^2-r3^2r4)→r7
88: if r7>0;√r7→r7
89: if not (sgn(T)→r8);1→r8
90: asn((r5-sgn(r8)r7)/r6)→p4;ret
91:
92: "Iter":fxd 0
93: 0→I→U;1969→V
94: "L1":(U+V)/2→P;cll 'pix'(T/2,P,A,N,R)
95: cll 'GP'(A,'dist'(N,D),L[3],L[1])
96: cll 'pixdir'(A,L[1],B);L[4]+asn(sin(B)sin(D)/cos(L[3]))→L[2]
97: cll 'equa'(L[1],L[2],S,O,M,K)
98: I+1→I;dsp "Iter:",I;if I>20;999→R
99: if abs(K-L→r0)<.00028 or R=999;gto +4
```

```
100: if r0<0;P→U
101: if r0>0;P→V
102: gto "L1"
103: ret
104:
105: "START":
106: fmt c35," = ",f15.4
107: beep;dsp "Key Special Function to Continue";stp
108:
109: "5":
110: "DegMin-DecDeg and ScanLine-Time conversions":
111: prt "Conversion Codes";fmt c10," =",c4
112: wrt 16,"seconds","S","scan lines","L","dec deg","DD","deg min","DM"
113: wrt 16,"QUIT","Q";spc 2
114: ent "Enter value and Identity code",C$
115: if pos(cap(C$),"Q");cll '----';gto "START"
116: val(C$)→C
117: fmt 1,10x,f7.2," SECONDS    =",f6.0," SCAN LINES"
118: fmt 2,10x,f7.0," SCAN LINES =",f7.2," SECONDS"
119: fmt 3,10x,f10.4," DEGREES MINUTES =",f10.4," DECIMAL DEGREES"
120: fmt 4,10x,f10.4," DECIMAL DEGREES =",f10.4," DEGREES MINUTES";fxd 1
121: if pos(cap(C$),"S");wrt F+.1,C,'scanli'(C)
122: if pos(cap(C$),"L");wrt F+.2,C,'second'(C)
123: if pos(cap(C$),"M");wrt F+.3,C,'dd'(C)
124: if pos(cap(C$),"DD");wrt F+.4,C,'dm'(C)
125: gto -11
126:
127: "6":
128: "Enter Equa crossing Lon & Time, Sta. Lat-Lon, get sta pix vals & time":
129: ent "Enter Equatorial crossing Long.",L
130: ent "Time or crossing    (HH.MMm)",Z
131: ent "Enter Station Latitiude",L[3],"Station Longitude",L[4],"Tilt",T
132: gsb "Iter"
133: wtb F,10,10
134: wrt F,"STATION LATITUDE",'dm'(L[3]),"STATION LONGITUDE",'dm'(L[4])
135: wrt F,"EQUATORIAL LONGITUDE",'dm'(L),"TIME AT EQUATOR",Z
136: wrt F,"TILT",T
137: cll '----';if R=999;wrt F,"STATION NOT SEEN THIS PASS, R",999;gto +4
138: wrt F,"TIME OF STATION PASSAGE",'dm'('dd'(Z)+'dd'(S/6000))
139: wrt F,"STATION PIXEL NUMBER",prnd(P,0)
140: wrt F,"SECONDS SINCE EQUATOR",S,"SCAN LINES SINCE EQUATOR",M
141: gto "START"
142:
143: "7":
144: "Enter Pixel Lat-Lon, Pixel #, get G.P. Lat-Lon, & Equa data":
145: ent "Enter Pixel lat.",L[3],"Pixel lon.",L[4],"Pixel #",P,"Tilt",T
146: cll 'pix'(T/2,P,A,N,R);cll 'dist'(N,D)
147: cll 'GP'(A,D,L[3],L[1])
148: cll 'pixair'(A,L[1],B);L[4]+asn(sin(B)sin(D)/cos(L[3]))→L[2]
149: cll 'equa'(L[1],L[2],S,O,M,L)
```

```
150: wtb F,10,10
151: wrt F,"PIXEL LATITUDE",'dm'(L[3]),"PIXEL LONGITUDE",'dm'(L[4])
152: wrt F,"PIXEL #",P,"TILT",T
153: cll '----';wrt F,"G.P. LATITUDE",'dm'(L[1])
154: wrt F,"G.P. LONGITUDE",'dm'(L[2]),"EQUATORIAL LONGITUDE",'dm'(L)
155: wrt F,"SECONDS SINCE EQUATOR",S,"SCAN LINES SINCE EQUATOR",M
156: gto "START"
157:
158:
159:
160: "8":
161: "Enter Pixel Lat-Lon, X-Y corrds, second pix X-Y coords,":
162: "Get second pixel Lat-Lon":
163: ent "Enter 1st Pixel lat.",L[3],"1st Pixel lon.",L[4],"1st Pixel #",P
164: ent "1st Pixel Scan line #",M[1],"Tilt",T
165: ent "Enter 2nd Pixel #",Q,"2nd Pixel Scan line #",M[2]
166: cll 'pix'(T/2,P,A,N,R);cll 'GP'(A,'dist'(N,D),L[3],L[1])
167: cll 'pixdir'(A,L[1],B);L[4]+asn(sin(B)sin(D)/cos(L[3]))→L[2]
168: cll 'equa'(L[1],L[2],S,O,M,L)
169: cll 'latlon'(S+'second'(M[1]-M[2]),L,L[1],L[2])
170: cll 'pix'(T/2,Q,A,N,R)
171: cll 'pixlat'(L[1],'dist'(N,D),'pixdir'(A,L[1],B),L[5])
172: cll 'pixlon'(L[5],L[2],D,B,L[6])
173: wtb F,10,10;wrt F,"FIRST PIXEL LAT",'dm'(L[3])
174: wrt F,"FIRST PIXEL LONG",'dm'(L[4]),"FIRST PIXEL #",P
175: wrt F,"FIRST PIXEL SCANLINE #",M[1],"SECOND PIXEL #",Q
176: wrt F,"SECOND PIXEL SCAN LINE #",M[2],"TILT",T;gsb "----"
177: wrt F,"SECOND PIXEL LAT",'dm'(L[5]),"SECOND PIXEL LONG",'dm'(L[6])
178: gto "START"
179:
180: "9":
181: "Enter Reference Pixel Lat-Lon, X-Y corrds, station pix Lat-Long,":
182: "Get station pixel X-Y values":
183: ent "Enter Ref Pixel lat.",L[5],"Ref Pixel lon.",L[6],"Ref Pixel #",Q
184: ent "Ref Pixel Scan line #",M[1],"Enter Tilt",T
185: ent "Enter Station Latitude",L[3],"Station Longitude",L[4]
186: cll 'pix'(T/2,Q,A,N,R);cll 'GP'(A,'dist'(N,D),L[5],L[1])
187: cll 'pixdir'(A,L[1],B);L[6]+asn(sin(B)sin(D)/cos(L[5]))→L[2]
188: cll 'equa'(L[1],L[2],S,O,M,L);M→M[2];gsb "Iter"
189: M[1]+(M[2]-M)→M;prnd(M,0)→M;prnd(P,0)→P
190: wtb F,10,10;wrt F,"REFERENCE PIXEL LATITUDE",'dm'(L[5])
191: wrt F,"REF PIXEL LONGITUDE",'dm'(L[6]),"REF PIXEL X VALUE",Q
192: wrt F,"REF PIXEL Y VALUE",M[1],"STATION LATITUDE",'dm'(L[3])
193: wrt F,"STATION LONGITUDE",'dm'(L[4]),"TILT",T;gsb "----"
194: wrt F,"STATION PIXEL X VALUE",P,"STATION PIXEL Y VALUE",M
195: gto "START"
196:
197:
198: "10":
199: "Enter G.P. Lat-Lon, Tilt, Pixel#; ret Pixel Lat-Lon":
```

```
200: ent "Enter G.P. Lat",L[1],"G.P. Lon",L[2],"Tilt",T,"Pixel#",P
201: cll 'pix'(T/2,P,A,N);cll 'dist'(N,D)

202: cll 'pixdir'(A,L[1],B);cll 'pixlat'(L[1],D,B,L[3])
203: cll 'pixlon'(L[3],L[2],D,B,L[4])
204: wtb F,10,10
205: wrt F,"G.P. LATITUDE",'dm'(L[1]),"G.P. LONGITUDE",'dm'(L[2]),"TILT",T
206: wrt F,"PIXEL #",P;cll '----'
207: wrt F,"NADIR ANGLE",N,"AZIMUTH ANGLE",A
208: wrt F,"DISTANCE TO PIXEL",D,"BEARING TO PIXEL",B
209: wrt F,"PIXEL LATITUDE",'dm'(L[3]),"PIXEL LONGITUDE",'dm'(L[4])
210: gto "START"
211:
212: "11":
213: "Change Roll, Pitch, and/or Yaw":
214: ent "Enter Roll",W,"Pitch",X,"Yaw",Y
215: fmt 2/,25x,"NEW ROLL, PITCH, YAW:";wrt F
216: fmt c35," = ",f7.2;wrt F,"ROLL",W,"PITCH",X,"YAW",Y
217: gsb "----"
218: gto "START"
219:
220: "Notes":
221: if F=701;fmt 3/;wrt 701
222: if F=6;wtb 6,102,125
223: fmt 7x,c
224: wrt F,"     THE CZNAV PROGRAM CAN BE USED TO CALCULATE A VARIETY OF"
225: wrt F,"CZCS NAVIGATION PROBLEMS.  THE ROUTINES ARE ACCESSED BY USE OF"
226: wrt F,"THE SPECIAL FUNCTION KEYS.  NOTE:  ALL ANGLES ARE ENTERED"
227: wrt F,"IN DECIMAL DEGREES, OUTPUT IS IN DEGREES MINUTES."
228: wrt F,"FOR DEGREES-MINUTES ENTRY, FUNCTION KEY F0 WILL CONVERT THE"
229: wrt F,"ANGLE AND ENTER IT; USE KEY F0 INSTEAD OF THE CONTINUE KEY."
230: wrt F,"TILT IS INITIALIZED TO 20; ROLL, PITCH, YAW TO 0.  TILT IS"
231: wrt F,"ASKED EACH TIME, USE F11 FOR R.P.Y.  VARIABLES ARE GLOBAL."
232: wtb F,10,10
233: fmt "KEY",30x,"DESCRIPTION",/;wrt F
234: fmt 0,"F",f2.0,":   ",c;fmt 1,7x,c;fxd 1
235: wrt F,6,"FROM EQUATORIAL CROSSING LONG, TIME, AND STATION LAT-LONG,"
236: wrt F+.1,"CALCULATE TIME OF STATION PASSAGE AND PIXEL #."
237: wrt F,7,"FROM PIXEL LAT-LONG AND PIXEL #, CALCULATE G.P. LAT-LONG."
238: wrt F,8,"FROM 1ST PIXEL LAT-LONG, X-Y VALUES, AND 2ND PIXEL X-Y VALUES,"
239: wrt F+.1,"CALCULATE 2ND PIXEL LAT-LONG."
240: wrt F,9,"FROM REF PIXEL LAT-LONG, X-Y VALUES, AND STA PIXEL LAT-LONG,"
241: wrt F+.1,"CALCULATE STATION PIXEL X-Y VALUES."
242: wrt F,10,"FROM G.P. LAT-LONG AND PIXEL #, CALCULATE PIXEL LAT-LONG."
243: wrt F,11,"ENTER NEW ROLL, PITCH, AND YAW VALUES."
244: wrt F,5,"DEG MIN - DEC DEG AND TIME - SCAN LINE CONVERSIONS."
245: wrt F,0,"DEGREES MINUTES ENTRY."
246: if F=701;wtb 701,12
247: ret
248:
```

```
249: "Variables":
250: "A = satellite azimuth to pixel":
251: "B = true bearing from G.P. to pixel":
252: "C = ":
253: "D = distance to pixel in degrees great circle arc":
254: "E = ":
255: "F = output device number":
256: "G = ":
257: "H = ":
258: "I = general counter":
259: "J = ":
260: "K = equatorial longitude used in 'Iter'":
261: "L = equatorial longitude of satellite crossing":
262: "M = scan lines from equator to G.P.":
263: "N = satellite nadir angle from G.P. to pixel":
264: "O = degrees great circle arc from equa. to G.P.":
265: "P = pixel number (primary or reference pixel)":
266: "Q = pixel number (secondary pixel)":
267: "R = satellite mirror shaft angle":
268: "S = seconds since equatorial crossing":
269: "T = satellite tilt angle (2 tau)":
270: "U = used in 'Iter'":
271: "V = used in 'Iter'":
272: "W = satellite roll angle":
273: "X = satellite pitch angle":
274: "Y = satellite yaw angle":
275: "Z = Time of equatorial crossing":
276: "L[*] = latitude-longitude array (odd=lat, even=lon)":
```

```
C     CZSUB.DOC - Documentation of the CZCS Navigation Subroutines
C
C     ********************************************************************
C                     CZSUB1 - CZCS NAVIGATION SUBR #1
C     Given a CZCS G.P. latitude, longitude and travel time in seconds,
C     this routine calculates a new G.P. latitude and longitude.
C
C        CALL CZSUB1 (XINCLP,PERIOD,XLATO,XLONGO,T,XLAT,XLONG,IERR)
C
C     ********************************************************************
C
C     ********************************************************************
C               CZSUB2 - CZCS Navigation Subr #2
C     Given the CZCS G.P. latitude, longitude, tilt, and scan angle,
C     this subroutine will return the pixel latitude and longitude.
C
C        CALL CZSUB2(XINCLP,ALTCOR,XLATO,XLONGO,TILT,XNU,XLAT,
C     * XLONG,PITCH,ROLL,YAW,IERR)
C
C     ********************************************************************
C
C     ********************************************************************
C               CZSUB3 - CZCS Navigation Subr #3
C     This subroutine returns the Latitude and Longitude of the Ground
C     Point of the Satellite.  The input variables are the Latitude and
C     Longitude, Tilt and Scan Angle of a Pixel, and the Roll, Pitch,
C     and Yaw, Satellite altitude factor, and equatorial trackline
C     inclination angle of the Satellite.  See CZSUB Variable index for
C     explanation of variable names.
C
C        CALL CZSUB3(XINCLP,ALTCOR,XLAT,XLONG,TILT,XNU,XLATO,
C     * XLONGO,PITCH,ROLL,YAW,IERR)
C
C     ********************************************************************
C
C     ********************************************************************
C                CZSUB4 - CZCS Navigation Subr #4
C     Given a satellite G.P. latitude and longitude,
C     this subroutine calculates the longitude of equatorial crossing,
C     and the satellite travel time from the equator to the G.P.
C
C        CALL CZSUB4 (XINCLP,PERIOD,XLATO,XLONGO,XLONGP,T,IERR)
C
C     ********************************************************************
C
C     ********************************************************************
```

```
C     ***********************************************************************
C                  CZSUB5 - CZCS Navigation Subr #5
C     Given a pixel latitude and longitude and an equatorial crossing
C     longitude, this subroutine iterates to find a satellite G.P.
C     latitude and longitude which (1) contains the pixel in its scan
C     and (2) crossed the equator at the given longitude.  Also returned
C     are the scan angle (XNU) and transit time (T).  If the iteration
C     fails, meaning no ground point meets the criteria, a value of -9999
C     is returned for XNU.
C
C        CALL CZSUB5(XINCLP,ALTCOR,PERIOD,XLONGP,XLAT,XLONG,TILT,
C     * XLATO,XLONGO,T,XNU,PITCH,ROLL,YAW,IERR)
C
C     ***********************************************************************
C
C     ***********************************************************************
C                  CZSUBA - CZCS Navigation subr A
C     This is an internal subroutine called by several of the other CZCS
C     navigation subroutines.  Given the tilt, scan angle, roll, pitch,
C     and yaw, this routine calculates the trackline relative bearing
C     to the pixel, the effective scan angle, and the earth surface
C     distance (in radians) from the satellite G.P. to the pixel.
C
C        SUBROUTINE CZSUBA(ALTCOR,TILT,XNU,RPSI,RTHS,RTH,PITCH,ROLL,YAW
C     *  ,IERR)
C
C     ***********************************************************************
C
C
C                        CZSUB - VARIABLES
C
C     These are the variables named in the call lists in the various CZCS
C     navigation subroutines.  All variables are REAL*8 except IERR.
C
C     ALTCOR      The altitude correction (R+h)/R, where R is the radius of
C                 the earth and h is the orbital altitude of the satellite.
C                 Supplied by the main or calling program.
C
C     IERR        This is a logical variable which when true enables error
C                 trace printouts at the terminal.  Supplied by the main or
C                 calling program.  LOGICAL*2
C
C     PERIOD      The orbital period of the satellite in seconds.  Supplied
C                 by the main or calling program.
C
C     PITCH       The state of rotation of the satellite about the Y axis
C                 in degrees.  Supplied by the main or calling program.
C
C     ROLL        The state of rotation of the satellite about the X axis
C                 in degrees.  Supplied by the main or calling program.
```

```
C
C    RPSI        The trackline bearing to the pixel in radians.  Returned
C                by CZSUBA.
C
C    RTH         The earth surface distance from the G.P. to the pixel in
C                radians.  Returned by CZSUBA.
C
C    RTHS        The effective scan angle (as modified by tilt) in radians.
C                Returned by CZSUBA.
C
C    T           The orbital travel time of the satellite between two
C                points in seconds.  For example the time for satellite
C                transit from the equator to the current G.P.  Supplied or
C                returned depending on subroutine.
C
C    TILT        The tilt angle of the CZCS scan view in degrees.  This
C                is as read from the satellite (or double the tilt angle
C                of the mirror shaft ,Tau).  Supplied by the main or
C                calling program.
C
C    XINCLP      The trackline equatorial inclination angle, 9.28 deg
C                west of north.
C
C    XLAT        The latitude of the pixel in decimal degrees.  Supplied or
C                returned.  Or, the new latitude in CZSUB1; returned.
C
C    XLATO       The latitude of the G.P. in decimal degrees.  Supplied or
C                returned.  Or, the input latitude in CZSUB1.
C
C    XLONG       The longitude of the pixel in decimal degrees.  Supplied
C                or returned.  The return longitude in CZSUB1.
C
C    XLONGO      The longitude of the G.P. in decimal degrees.  Supplied
C                or returned.  The input longitude in CZSUB1.
C
C    XLONGP      The longitude of equatorial crossing in decimal degrees.
C                supplied or returned.
C
C    XNU         The mirror scan angle in degrees.  Supplied by main or
C                calling program.
C
C    YAW         The state of rotation of the satellite about the Z axis
C                in degrees.  Supplied by the main or calling program.
C
C    Note that while the angular variables are supplied or returned
C    in degrees, the internal computer computations are performed
C    in radians.  These angular variables are converted in the
C    subroutines and then reconverted before return.  The exception
C    to this is CZSUBA whose call list contains radians.  Variables
C    which start with an X are in degrees and those with an R
```

```
C     are in radians.  For example within CZSUB3, XNU is converted to
C     RNU and XLAT becomes RLAT.


C-------------------------------------------------------------
C *** LIST OF SUBPROGRAMS IN FILE "CZSUBS":     (CZSUB1 - CZSUBA)
C        "       CZSUB1
C        "       CZSUB2
C        "       CZSUB3
C        "       CZSUB4
C        "       CZSUB5
C        "       CZSUBA
C                      *** END DIRECTORY ***
C-------------------------------------------------------------
C$
C     ********************************************************************
C                     CZSUB1 - CZCS NAVIGATION SUBR #1
C     Given a CZCS G.P. latitude, longitude and travel time in seconds,
C     this routine calculates a new G.P. latitude and longitude.
C
C         22 July 1981, W. Wilson, Visibility Laboratory,
C         Scripp's Institution of Oceanography.
C
C     ********************************************************************
C
      SUBROUTINE CZSUB1 (XINCLP,PERIOD,XLATO,XLONGO,T,XLAT,XLONG,IERR)
C
         IMPLICIT REAL*8(A-H,O-Z)
        REAL*8 PI,PID2,DEGRD,DATAN,DATAN2,DSIN,DCOS,DARCOS,DARSIN,DSQRT
        REAL*8 RINCLP,RLATO,RINCLO,DT,X,RLAM,DTAN,DABS,RLAT
        LOGICAL IERR
C
        PI = 4.0*DATAN(1.0D0)
        DEGRD = PI/180.0
C
       RINCLP = XINCLP*DEGRD            /* equatorial inclination
       RLATO = XLATO*DEGRD
C
C             Calculate trackline inclination at initial G.P.
       RINCLO = DASIN(DSIN(RINCLP)/DCOS(RLATO))
C
C             Calculate degrees trackline traveled in T (fixed earth)
       DT = T*360.0/PERIOD*DEGRD
C
C             Calculate new G.P. latitude
       X = DCOS(DT)*DSIN(RLATO)+DSIN(DT)*DCOS(RLATO)*DCOS(RINCLO)
       RLAT = DASIN(X)
       XLAT = RLAT/DEGRD
C
```

```
C             Calculate delta longitude
      RLAM = DASIN(DSIN(RINCLO)/DCOS(RLAT)*DSIN(DT))
       DLAM = RLAM/DEGRD
C
C             New longitude = old long + del lon + earth rotation
       XLONG = XLONGO + DLAM + T*4.167E-3
C
       IF(IERR) PRINT 2000,XLATO,XLONGO,XLAT,XLONG,DLAM,T
 2000  FORMAT(' CZSUB1 - XLATO,XLONGO,XLAT,XLONG,DLAM,T = ',/,
     * 1P6G13.6)
       RETURN
C              *** END SUBROUTINE CZSUB1 ***
       END
C$
C    ************************************************************************
C             CZSUB2 - CZCS Navigation Subr #2
C    Given the CZCS G.P. latitude, longitude, tilt, and scan angle,
C    this subroutine will return the pixel latitude and longitude.
C
C        22 July 1981, W. Wilson, Visibility Laboratory,
C        Scripp's Institution of Oceanography.
C
C    ************************************************************************
C
       SUBROUTINE CZSUB2(XINCLP,ALTCOR,XLATO,XLONGO,TILT,XNU,XLAT,
     * XLONG,PITCH,ROLL,YAW,IERR)
C
        IMPLICIT REAL*8(A-H,O-Z)
       REAL*8 PI,PID2,DEGRD
       REAL*8 RINCLP,RLATO,RINCLO,RPSIP,RPSI,RTH,X,RLAT,DLAM
       REAL*8 RTHS
C
       LOGICAL IERR
C
       PI = 4.0*DATAN(1.0D0)
       DEGRD = PI/180.0
C
       IF(IERR) PRINT 2000
 2000  FORMAT(' ENTER CZSUB2')
C
C             Get Psi and Theta from CZSUBA
       CALL CZSUBA(ALTCOR,TILT,XNU,RPSI,RTHS,RTH,PITCH,ROLL,YAW,IERR)
C
       RINCLP = XINCLP*DEGRD
       RLATO = XLATO*DEGRD
C
C             Calculate trackline inclination at G.P. latitude
       RINCLO = DASIN(DSIN(RINCLP)/DCOS(RLATO))
C             Calculate true bearing to pixel
       RPSIP = RPSI - RINCLO
```

```
C             Calculate pixel latitude
        X = DSIN(RLATO)*DCOS(RTH) + DCOS(RLATO)*DSIN(RTH)*DCOS(RPSIP)
        RLAT = DASIN(X)
        XLAT = RLAT/DEGRD                /* convert back to degrees
C
C             Calculate delta longitude
        X = DSIN(RPSIP)/DCOS(RLAT)*DSIN(RTH)
        DLAM = DASIN(X)
        XDLAM = DLAM/DEGRD
C             Calculate pixel longitude
        XLONG = XLONGO - XDLAM
C
        PSIP = RPSIP/DEGRD               /* convert back to degrees
        TH = RTH/DEGRD
        IF(IERR) PRINT 2003,PSIP,XLAT,XDLAM
 2003  FORMAT(' PSIP,XLAT,XDLAM = ',1P4G13.6)
C
        RETURN
C               *** END SUBROUTINE CZSUB2 ***
        END
C$
C    ****************************************************************
C                         SUBROUTINE CZSUB3
C    This subroutine returns the Latitude and Longitude of the Ground
C    Point of the Satellite.  The input variables are the Latitude and
C    Longitude, Tilt and Scan Angle of a Pixel, and the Roll, Pitch,
C    and Yaw, Satellite altitude factor, and equatorial trackline
C    inclination angle of the Satellite.
C
C        22 July 1981, W. Wilson, Visibility Laboratory,
C        Scripp's Institution of Oceanography.
C
C    ****************************************************************
C
      SUBROUTINE CZSUB3(XINCLP,ALTCOR,XLAT,XLONG,TILT,XNU,XLATO,
     * XLONGO,PITCH,ROLL,YAW,IERR)
C
        IMPLICIT REAL*8(A-H,O-Z)
        REAL*8 PI,PID2,DEGRD
        REAL*8 RPSI,RTH,RLAT,RINCLP,A,B,C,D,AB,XD,X,XP,X1,X2
        REAL*8 RLATO1,RLATO2,RLATO,RINCLO,RPSIP,DLAM
        REAL*8 RTHS
        LOGICAL IERR
C
        PI = 4.0*DATAN(1.0D0)
        DEGRD = PI/180.0                 /* Degrees-Radians conv
C
       IF(IERR) PRINT 2000
 2000  FORMAT(' ENTER CZSUB3')
        CALL CZSUBA(ALTCOR,TILT,XNU,RPSI,RTHS,RTH,PITCH,ROLL,YAW,IERR)
```

```
        RLAT = XLAT*DEGRD
        RINCLP = XINCLP*DEGRD
C
C             Set up Quadratic variables from CZCS angular components.
C             See documentation for derivation.
C
        A = DSIN(RLAT) - DSIN(RTH)*DSIN(RPSI)*DSIN(RINCLP)
        B = DCOS(RTH)
        C = DSIN(RTH)*DCOS(RPSI)
        D = DCOS(RINCLP)**2
C
        AB = A*B                          /* Calculate intermediate vals
        XD = B**2 + C**2
        IF(IERR) PRINT 2001,A,B,C,D,XD,AB
        X = AB**2 - XD*(A**2-D*C**2)
        XP = 0.0
        IF(X.GT.1.E-10) XP = DSQRT(X)
 2001   FORMAT(' A,B,C,D,XD,AB = ',1P6G13.6)
C
        X1 = (AB + XP)/XD                 /* The +/- values from the
        X2 = (AB - XP)/XD                 /* Quadratic.
      IF(IERR) PRINT 2005,X,X1,X2
 2005 FORMAT(' X,X1,X2-',1P3G13.6)
        RLAT01 = DASIN(X1)
        RLAT02 = DASIN(X2)
      IF(IERR) PRINT 2006,RLAT01,RLAT02
 2006 FORMAT(' RLAT01,RLAT02-',1P2G13.6)
C
C             If the tilt is positive choose the negative quad.
C             component and if negative the positive.
        RLATO = RLAT02
        IF(TILT.LT.0.0) RLATO = RLAT01
        XLAT01 = RLAT01/DEGRD
        XLAT02 = RLAT02/DEGRD
        XLATO = RLATO/DEGRD
C
C             Calculate true bearing to pixel
        RINCLO = DASIN(DSIN(RINCLP)/DCOS(RLATO))
        RPSIP = RPSI - RINCLO
      IF(IERR) PRINT 2002,RPSIP,RTH,RLAT
 2002 FORMAT(' RPSIP,RTH,RLAT-',1P3G13.6)
C
C             Calculate longitude difference G.P. to Pix
        X = DSIN(RPSIP)*DSIN(RTH)/DCOS(RLAT)
        DLAM = DASIN(X)
        XLAM = DLAM/DEGRD
        XLONGO = XLONG + XLAM             /* Calc G.P. Longitude
C
        IF(IERR) PRINT 2003,XLAT01,XLAT02,XLONGO,XLAM
 2003   FORMAT(' LAT1,LAT2,LONG,DLONG = ',1P4G13.6)
```

```
        PSIP = RPSIP/DEGRD
        IF(IERR) PRINT 2004,PSIP,XLATO,XLAM
 2004   FORMAT(' PSIP,XLAT,XDLAM = ',1P4G13.6)
        RETURN
C               *** END SUBROUTINE CZSUB3 ***
        END
C    ***************************************************************
C               CZSUB4 - CZCS Navigation Subr #4
C    Given a satellite G.P. latitude and longitude,
C    this subroutine calculates the longitude of equatorial crossing,
C    and the satellite travel time from the equator to the G.P.
C
C        22 July 1981, W. Wilson, Visibility Laboratory,
C        Scripp's Institution of Oceanography.


C
C      ************************************************************
C
        SUBROUTINE CZSUB4 (XINCLP,PERIOD,XLATO,XLONGO,XLONGP,T,IERR)
         IMPLICIT REAL*8(A-H,O-Z)
C
        REAL*8 PI,PID2,DEGRD,DATAN,DATAN2,DSIN,DCOS,DARCOS,DARSIN,DSQRT
        REAL*8 RINCLP,RLATO,RLAM,RT,DTAN,DABS,DT
C
        LOGICAL IERR
C
        PI = 4.0*DATAN(1.0D0)
        DEGRD = PI/180.0
C
        RINCLP = XINCLP*DEGRD
        RLATO = XLATO*DEGRD
C
C             Calculate the delta longitude
        RLAM = DASIN(DTAN(RLATO)*DTAN(RINCLP))
C
C             Calculate the degrees traveled and transit time
        RT = DASIN(DSIN(RLATO)/DCOS(RINCLP))
        DT = RT/DEGRD
        T = DT*PERIOD/360.0
C
C             Calculate the equatorial crossing longitude
        DLAM = RLAM/DEGRD
        XLONGP = XLONGO - DLAM - T*4.167E-3
C
        IF(IERR) PRINT 2000,XLATO,XLONGO,XLONGP,DT,DLAM,T
 2000   FORMAT(' CZSUB4 - XLATO,XLONGO,XLONGP,DT,DLAM,T = ',/,
     * 1P6G13.6)
C
        RETURN
```

```
C               *** END SUBROUTINE CZSUB4 ***
      END
C$
C    ***********************************************************************
C                CZSUB5 - CZCS Navigation Subr #5
C    Given a pixel latitude and longitude and an equatorial crossing
C    longitude, this subroutine iterates to find a satellite G.P.
C    latitude and longitude which (1) contains the pixel in its scan
C    and (2) crossed the equator at the given longitude.  Also returned
C    are the scan angle (XNU) and transit time (T).  If the iteration
C    fails, meaning no ground point meets the criteria, a value of
C    INTL(-9998) is returned for XNU.
C
C        22 July 1981, W. Wilson, Visibility Laboratory,
C        Scripp's Institution of Oceanography.
C
C    ***********************************************************************
C
      SUBROUTINE CZSUB5(XINCLP,ALTCOR,PERIOD,XLONGP,XLAT,XLONG,TILT,
     * XLATO,XLONGO,T,XNU,PITCH,ROLL,YAW,IERR)
C
       IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL IERR
C
      XL = 0.0                          /* initial pixel limits
      XH = 1969.
      ITER = 0
C
C            Start with a known pixel lat-lon ........
C
   10 CONTINUE
      XM = (XL+XH)/2.0                  /* estimate pixel number
      XNU = -39.36 + (XM-1.0)*0.04      /* and scan angle
C
C            Go calculate a coresponding G.P. lat-lon
C
      CALL CZSUB3 (XINCLP,ALTCOR,XLAT,XLONG,TILT,XNU,XLATO,
     * XLONGO,PITCH,ROLL,YAW,IERR)
C
C            Go find the estimated G.P.'s equatorial crossing lon
C
      CALL CZSUB4(XINCLP,PERIOD,XLATO,XLONGO,XLONGQ,T,IERR)
C
C            Compare estimated eq. lon with input eq. lon
C
      DIF = XLONGQ - XLONGP
      IF(DABS(DIF).LT..00028) GO TO 100     /* good - go return
      ITER = ITER + 1
      IF(ITER.GT.20) GO TO 200          /* won't work - quit
C            or
```

```
        IF(DIF.GT.0.0) XH = XM          /* adjust scan limits
        IF(DIF.LT.0.0) XL = XM          /* for next try
C
        IF(IERR) PRINT 2000,XM,XNU,XLATO,XLONGO,XLONGP,XLONGQ,DIF
 2000  FORMAT(' CZSUB5 - XM,XNU,XLATOMXLONGO,XLONGP,XLONGQ,DIF =',
     * /,F7.1,F8.2,1P6G11.5)
C
        GO TO 10                        /* go try again
C
  100 CONTINUE
        IF(IERR) PRINT 2000,XM,XNU,XLATO,XLONGO,XLONGP,XLONGQ,DIF
        RETURN
  200  CONTINUE
         PRINT 2001
 2001  FORMAT(' FAILED TO CONVERGED - PROBABLY OUTSIDE IMAGE')
        XNU = -9999.
         RETURN
C               *** END SUBROUTINE CZSUB5 ***
        END
C$
C    ********************************************************************
C                    CZSUBA - CZCS Navigation subr A
C    This is an internal subroutine called by several of the other CZCS
C    navigation subroutines.  Given the tilt, scan angle, roll, pitch,
C    and yaw, this routine calculates the trackline relatlive bearing
C    to the pixel, the effective scan angle, and the earth surface
C    distance (in radians) from the satellite G.P. to the pixel.
C
C        22 July 1981, W. Wilson and J. Nolten, Visibility Laboratory,
C        Scripp's Institution of Oceanography.
C
C    ********************************************************************
C
       SUBROUTINE CZSUBA(ALTCOR,TILT,XNU,RPSI,RTHS,RTH,PITCH,ROLL,YAW
     *  ,IERR)
C
        IMPLICIT REAL*8(A-H,O-Z)
       REAL*8 PI,PID2,DEGRD,IROOT
       REAL*8 PI3D2,RPSI,RTH,RNU,RTHS,RTILT
C
       LOGICAL IERR
C
C
       PID2 = DATAN(1.0D0)*2.0
       PI = PID2*2.0                    /* calculate pi
       PI3D2 = PID2*3.0
C
       DEGRD = PI/180.0                 /* degree - radian conversion
C
      IF(IERR) PRINT 2000,ALTCOR,TILT,XNU
```

```
 2000  FORMAT(' ENTER CZSUBA , ALTCOR,TILT,NU = ',3F8.3)
       RNU = XNU*DEGRD
        T = TILT/2.0
        RT = T*DEGRD
       IF(TILT.NE.0.0) GO TO 1
       RPSI = PID2
       IF(XNU.LT.0.0) RPSI = PI3D2
       RTHS = DABS(RNU)
       GO TO 2
    1  CONTINUE
         IF(IERR) PRINT 3001,RT,RNU
 3001  FORMAT(' 1',' RT,RNU',1P3G13.6)
C
C               Calculate Theta-n and Phi-n
C
         IROOT = 1.0D0/ DSQRT(2.0D0)
         RTHR = 2.*DACOS(IROOT*(DSIN(RT)*DCOS(RNU)+DCOS(RT)))
         RPHIM = DATAN(DSIN(RNU)/(DSIN(RT)-DCOS(RT)*DCOS(RNU)))+PI
         IF(IERR) PRINT 3002,RTHM,RPHIM,RTHR
 3002  FORMAT(' 2',' RTHM,RPHIM,RTHR',1P3G13.6)
C
C              Calculate Theta and Phi
C
         RTHP  = 1.0D0/DSQRT(2.0D0)
         RPHIP = DATAN(DTAN(RTHR)*DSIN(RPHIM))
         IF(TILT.LT.0.0) RPHIP = PI + RPHIP
         IF(IERR) PRINT 3003,RTHP,RPHIP
 3003  FORMAT(' 3',' RTHP,RPHIP',1P2G13.6)
C
      IF (ROLL.NE.0. .OR. PITCH.NE.0. .OR. YAW.NE.0.) GO TO 3
      RPSI = RPHIP
      RTHS = RTHP
      GO TO 2
   3  CONTINUE
C
C               Correct for roll pitch and yaw
C
         U = DSIN(RTHP)*DCOS(RPHIP)
         V = DSIN(RTHP)*DSIN(RPHIP)
         W = DCOS(RTHP)
         Y = YAW*DEGRD
         R = ROLL*DEGRD
         P = PITCH*DEGRD
        XP = (1.0D0 + Y*R*P)*U + (-Y+R*P)*V + P*W
        YP = Y*U + V - R*W
        ZP = (-P+Y*R)*U + (-P*Y+R)*V + W
         IF(IERR) PRINT 3004,U,V,W,XP,YP,ZP
 3004  FORMAT(' 4',' U,V,W,XP,YP,ZP',1P6G13.6)
         RTHS = DACOS(ZP)
         RPSI = DATAN2(YP,XP)
```

```
    2  CONTINUE
         RTH = DASIN(ALTCOR*DSIN(RTHS)) - RTHS
          THS = RTHS/DEGRD
         TH = RTH/DEGRD
         PSI = RPSI / DEGRD
         IF(IERR) PRINT 2001,THS,TH,PSI
 2001  FORMAT(' THS,TH,PSI = ',1P3G13.6)
C
         RETURN
C               *** END SUBROUTINE CZSUBA ***
         END
C$



C             ***CZCS NAVIGATION PROGRAM***
C
      PROGRAM CZNAV
C
C           This program uses the CZCS Navigation Subroutines to perform
C           a variety of CZCS location calculations.  A menu of functions
C           is provided, variables are global so that, if iterating, only
C           changing parameters need be reentered.  Output is displayed
C           on the terminal and written to an output file 'CZCS.DATA'.
C             22 July 1981, J. Nolten, Visibility Laboratory, Scripp's
C             Institution of Oceanography
C
      IMPLICIT REAL*8 (A-G,N-Z)
      INTEGER*4 PTR,TTY,CHOICE
      CHARACTER*80 CHAR
      LOGICAL IERR
         DATA IERR/.FALSE./,PTR/16/,TTY/1/
C
C        /* SETUP
      PER = 104.09                  /* SATELLITE CONSTANTS
      PERIOD = PER*60.
      XINCLP = 9.28
C     ALTCOR = (952. + 6371.)/ 6371.
      ALTCOR = 1.1494
      REV = 1./.12375
      ROLL = 0.
      PITCH = 0.
      YAW = 0.
      TILT = 20.
      ZERO = 0.
      NADIR = 0.32
C
      OPEN (UNIT    = PTR,
     *      FILE    = 'CZCS.DATA',
     *      STATUS  = 'UNKNOWN',
```

```
     *       ACCESS   = 'SEQUENTIAL',
     *       FORM     = 'UNFORMATTED',
     *       ERR      = 99)
C
 1000  FORMAT (///,15X,'Welcome to the CZCS Navigation program.',/)
 1010  FORMAT (5X,'This program provides a variety of CZCS calculations',
     *   /,'selectable from the menu to follow.  Besides the terminal,',
     *   /,'input and output are written to file "CZCS.DATA".')
 1020  FORMAT (5X,'Note:  ROLL,PITCH, and YAW are initialized to zero,',
     *   /,13X,'and TILT is initialized to +20.',/)
 1030  FORMAT (5X,'Please enter Date and/or any information you would',/,
     *  'like included as header on your output file. (80 char. avail)')
 1040  FORMAT (A)
       WRITE (TTY,1000)
       WRITE (TTY,1010)
       WRITE (TTY,1020)
       WRITE (TTY,1030)
       READ  (TTY,1040) CHAR
 2000  FORMAT (//,15X,'CZCS Navigation Calculations',/)
 2010  FORMAT (2X,A,//)
 2020  FORMAT (5X,50('-'),/)
       WRITE (PTR,2000)
       WRITE (PTR,2010) CHAR
       WRITE (PTR,1020)
       WRITE (PTR,2020)
C
C              /* MAIN LOOP
C
   10  CONTINUE
 1050  FORMAT (//,20X,'MENU',/)
 1055  FORMAT (' Angle Conventions: entry and output in DDD.MMmm',/,
     *         '                          North and West positive',/)
 1060  FORMAT (' 1:     Enter Pixel Lat-Lon, X-Y coords, get G.P. ',
     *   'Lat-Lon, Equator data.',/,
     *   ' 2:     Enter G.P. Lat-Lon, Pixel #; get Pixel Lat-Lon.',/,
     *   ' 3:     Enter Pixel Lat-Lon, X-Y coords, second Pixel X-Y,',
     *   /,8X,'get second Pixel Lat-Lon.',/,
     *   ' 4:     Enter Equatorial crossing Longitude, Time, ',
     *   'Station Lat-Lon;',/,8X,
     *   'get Station Pixel values and Station Time.',/,
     *   ' 5:     Enter Station Lat-Lon, Calculate Equtorial crossing',
     *   ' window.',/,
     *   ' 6:     Enter Pixel Lat-Lon, X-Y coords, Station Pixel ',
     *   'Lat-Lon,',/,8X,'get Station Pixel X-Y coords.',/,
     *   ' 7:     DegMin-DecDeg and Time-Scan Lines conversions',
     *   /,' 8:     Change Roll, Pitch, and Yaw settings.',
     *   /,' 9:     QUIT')
 1070  FORMAT (/,'PLEASE ENTER NUMBER OF CALCULATION TO PERFORM.')
       WRITE (TTY,1050)
       WRITE (TTY,1055)
```

```
      WRITE (TTY,1060)
      WRITE (TTY,1070)
      READ  (TTY,*) CHOICE
C
      GO TO (20,30,40,50,60,70,80,90,100),CHOICE
      WRITE (TTY,*) 'INVALID CALCULATION TYPE'
      GO TO 10
C
 4000  FORMAT (//,' INPUT')
 4001  FORMAT (20X,10('-'),/,' OUTPUT')
 4002  FORMAT (/,10X,30('-'),/)
 4010  FORMAT (' Pixel Latitude, Longitude:',2F10.4)
 4011  FORMAT (' First Pixel:',/,5X,'Latitude and Longitude:',
     *   2F10.4,/,5X,'X and Y values:',2I6)
 4013  FORMAT (' Tilt =',F7.1)
 4020  FORMAT (' Pixel Number:',I5,', Tilt:',F5.1)
 4030  FORMAT (' G.P.  Latitude, Longitude:',2F10.4)
 4040  FORMAT (' Equatorial crossing Longitude:',F10.4,/,
     *      ' Travel time since equator:',F7.1,' seconds.')
 4050  FORMAT (' Equatorial crossing Longitude:',F10.4)
 4051  FORMAT (' Equatorial crossing time:',F7.2)
 4052  FORMAT (' Station not seen by this Satellite pass.')
 4053  FORMAT (' Station Latitude and Longitude:',2F10.4)
 4054  FORMAT (' Optimal equatorial window from',F9.4,' to',
     *   F9.4,'.')
 4055  FORMAT (' Station Pixel X and Y values:',2I6)
 4060  FORMAT (' Second Pixel Latitude, Longitude:',2F10.4)
 4061  FORMAT (' Travel time between scans:',F7.1,' seconds.')
 4062  FORMAT (' Second Pixel X and Y values:',2I6)
 4070  FORMAT (' Time of Station passage:',F8.3,', Pixel #',I5)
 4090  FORMAT (' Roll, Pitch, Yaw changed to:',3F8.3)
 5000  FORMAT ()
C
C             From Pixel info, calculate G.P. info
C
   20  CONTINUE
 3000  FORMAT('Enter Pixel Latitude, Longitude, Pixel Number, and Tilt')
       WRITE (TTY,3000)
       READ  (TTY,*) XLAT,XLONG,PIXNO1,TILT
       XLAT = DMTOD(XLAT)
       XLONG = DMTOD(XLONG)
       XNU = -39.04 + .04*PIXNO1
         CALL CZSUB3(XINCLP,ALTCOR,XLAT,XLONG,TILT,XNU,XLATO,
     * XLONGO,PITCH,ROLL,YAW,IERR)
         CALL CZSUB4 (XINCLP,PERIOD,XLATO,XLONGO,XLONGP,T,IERR)
       XLAT = DTODM(XLAT)
       XLONG = DTODM(XLONG)
       XLATO = DTODM(XLATO)
       XLONGO = DTODM(XLONGO)
       XLONGP = DTODM(XLONGP)
```

```
C
      WRITE (TTY,4000)
      WRITE (TTY,4010) XLAT,XLONG
      WRITE (TTY,4020) PIXNO1,TILT
      WRITE (TTY,4001)
      WRITE (TTY,4030) XLATO,XLONGO
      WRITE (TTY,4040) XLONGP,T
      WRITE (TTY,4002)
C
      WRITE (PTR,4000)
      WRITE (PTR,4010) XLAT,XLONG
      WRITE (PTR,4020) PIXNO1,TILT
      WRITE (PTR,4001)
      WRITE (PTR,4030) XLATO,XLONGO
      WRITE (PTR,4040) XLONGP,T
      WRITE (PTR,4002)
C
      WRITE (TTY,*) 'Use RETURN key to continue program when ready.'
      READ  (TTY,5000)
      GO TO 10
C
C             From G.P. Lat-Lon and Pix #, Calculate Pixel Lat-Lon
C
   30  CONTINUE
 3010  FORMAT ('Enter G.P. Latitude, Longitude, Pixel #, and Tilt')
      WRITE (TTY,3010)
      READ  (TTY,*) XLATO,XLONGO,PIXNO1,TILT
      XLATO = DMTOD(XLATO)
      XLONGO = DMTOD(XLONGO)
      XNU = -39.04 + .04*PIXNO1
        CALL CZSUB2(XINCLP,ALTCOR,XLATO,XLONGO,TILT,XNU,XLAT,
     * XLONG,PITCH,ROLL,YAW,IERR)
      XLAT = DTODM(XLAT)
      XLONG = DTODM(XLONG)
      XLONGO = DTODM(XLONGO)
      XLATO = DTODM(XLATO)
      WRITE (TTY,4000)
      WRITE (TTY,4030) XLATO,XLONGO
      WRITE (TTY,4020) PIXNO1,TILT
      WRITE (TTY,4001)
      WRITE (TTY,4010) XLAT,XLONG
      WRITE (TTY,4002)
C
      WRITE (PTR,4030) XLATO,XLONGO
      WRITE (PTR,4020) PIXNO1,TILT
      WRITE (PTR,4001)
      WRITE (PTR,4010) XLAT,XLONG
      WRITE (PTR,4002)
C
      WRITE (TTY,*) 'Use RETURN key to continue program when ready.'
```

```
      READ  (TTY,5000)
      GO TO 10
C
C             From Lat-Lon and X-Y for Pixel #1 and X-Y for Pixel #2
C             Calculate Lat-Lon for Pixel #2
C
   40  CONTINUE
 3020  FORMAT ('Enter Pixel #1 Latitude, Longitude, Pixel and Scan #s')
 3021  FORMAT ('Enter Pixel and Scan #s for Pixel #2, and Tilt')
      WRITE (TTY,3020)
      READ  (TTY,*) XLAT,XLONG,PIXNO1,SCANO1
      XLAT = DMTOD(XLAT)
      XLONG = DMTOD(XLONG)
      WRITE (TTY,3021)
      READ  (TTY,*) PIXNO2,SCANO2,TILT
      XNU = -39.04 + .04*PIXNO1
      T = (SCANO1-SCANO2)*0.12375
        CALL CZSUB3(XINCLP,ALTCOR,XLAT,XLONG,TILT,XNU,XLATO,
     * XLONGO,PITCH,ROLL,YAW,IERR)
      WRITE (TTY,4000)
      WRITE (TTY,4011)DTODM(XLAT),DTODM(XLONG),PIXNO1,SCANO1
      WRITE (PTR,4000)
      WRITE (PTR,4011) DTODM(XLAT),DTODM(XLONG),PIXNO1,SCANO1
      CALL CZSUB1 (XINCLP,PERIOD,XLATO,XLONGO,T,XLAT,XLONG,IERR)

      XLATO = XLAT
      XLONGO = XLONG
      XNU = -39.04 + .04*PIXNO2
        CALL CZSUB2(XINCLP,ALTCOR,XLATO,XLONGO,TILT,XNU,XLAT,
     * XLONG,PITCH,ROLL,YAW,IERR)
C
      XLAT = DTODM(XLAT)
      XLONG = DTODM(XLONG)
      XLATO = DTODM(XLATO)
      XLONGO = DTODM(XLONGO)
C
      WRITE (TTY,4062) PIXNO2,SCANO2
      WRITE (TTY,4013) TILT
      WRITE (TTY,4001)
      WRITE (TTY,4060) XLAT,XLONG
      WRITE (TTY,4061) T
      WRITE (TTY,4002)
      WRITE (PTR,4062) PIXNO2,SCANO2
      WRITE (PTR,4013) TILT
      WRITE (PTR,4001)
      WRITE (PTR,4060) XLAT,XLONG
      WRITE (PTR,4061) T
      WRITE (PTR,4002)
C
      WRITE (TTY,*) 'Use RETURN key to continue program when ready.'
```

```
      READ  (TTY,5000)
      GO TO 10
C
C           Given Eq crossing lon and Sta. Lat-Lon,
C           Calc time of sta passage, scan posit
C
   50  CONTINUE
 3050  FORMAT ('Enter Equatorial Crossing Longitude, Time (HH.MMm)')
 3051  FORMAT ('Enter Station Latitude, Longitude, and Tilt')
      WRITE (TTY,3050)
      READ  (TTY,*) XLONGP,EQT
      XLONGP = DMTOD(XLONGP)
      WRITE (TTY,3051)
      READ  (TTY,*) XLAT,XLONG,TILT
      XLAT = DMTOD(XLAT)
      XLONG = DMTOD(XLONG)
        CALL CZSUB5(XINCLP,ALTCOR,PERIOD,XLONGP,XLAT,XLONG,TILT,
     * XLATO,XLONGO,T,XNU,PITCH,ROLL,YAW,IERR)
C
      STT = DTODM(DMTOD(EQT) + DMTOD(T/6000.))
      XLONGP = DTODM(XLONGP)
      XLAT = DTODM(XLAT)
      XLONG = DTODM(XLONG)
      PIXNOS = (XNU + 39.04)/.04
C
      WRITE (TTY,4000)
      WRITE (TTY,4053) XLAT,XLONG
      WRITE (TTY,4050) XLONGP
      WRITE (TTY,4051) EQT
      WRITE (TTY,4001)
      IF (INTL(XNU).EQ.-9998.) THEN
         WRITE (TTY,4052)
         GO TO 51
      END IF
      WRITE (TTY,4070) STT,PIXNOS
   51  CONTINUE
      WRITE (TTY,4002)
      WRITE (PTR,4000)
      WRITE (PTR,4053) XLAT,XLONG
      WRITE (PTR,4050) XLONGP
      WRITE (PTR,4051) EQT
      WRITE (PTR,4001)
      IF (INTL(XNU).EQ.-9998.) THEN
         WRITE (PTR,4052)
         GO TO 52
      END IF
      WRITE (PTR,4070) STT,PIXNOS
   52  CONTINUE
      WRITE (PTR,4002)
C
```

```
      WRITE (TTY,*) 'Use RETURN key to continue program when ready.'
      READ  (TTY,5000)
      GO TO 10
C
C             From Station Latitude, Longitude, Calc Equatorial window
C
   60 CONTINUE
      WRITE (TTY,3051)
      READ  (TTY,*) XLATO,XLONGO,TILT
      XLATO = DMTOD(XLATO)
      XLONGO = DMTOD(XLONGO)
        CALL CZSUB4 (XINCLP,PERIOD,XLATO,XLONGO,XLONGP,T,IERR)
        CALL CZSUB2(XINCLP,ALTCOR,ZERO,XLONGP,TILT,35.0D0,XLAT,
     * XLONGE,PITCH,ROLL,YAW,IERR)
        CALL CZSUB2(XINCLP,ALTCOR,ZERO,XLONGP,TILT,-35.0D0,XLAT,
     * XLONGW,PITCH,ROLL,YAW,IERR)
      XLONGP = DTODM(XLONGP)
      XLONGE = DTODM(XLONGE)
      XLONGW = DTODM(XLONGW)
      XLAT = DTODM(XLAT)
      XLATO = DTODM(XLATO)
      XLONGO = DTODM(XLONGO)
C
      WRITE (TTY,4000)
      WRITE (TTY,4053) XLATO,XLONGO
      WRITE (TTY,4001)
      WRITE (TTY,4040) XLONGP,T
      WRITE (TTY,4054) XLONGE,XLONGW
      WRITE (TTY,4002)
C
      WRITE (PTR,4000)
      WRITE (PTR,4053) XLATO,XLONGO
      WRITE (PTR,4001)
      WRITE (PTR,4040) XLONGP,T
      WRITE (PTR,4054) XLONGE,XLONGW
      WRITE (PTR,4002)
C
      WRITE (TTY,*) 'Use RETURN key to continue program when ready.'
      READ  (TTY,5000)
      GO TO 10
C
C             From reference pixel and station Lat-Lon get sta. X-Y
C
   70 CONTINUE
      WRITE (TTY,3020)
      READ  (TTY,*) XLAT,XLONG,PIXNO1,SCANO1
      XLAT = DMTOD(XLAT)
      XLONG = DMTOD(XLONG)
      WRITE (TTY,3051)
      READ  (TTY,*) SLAT,SLONG,TILT
```

```
      SLAT = DMTOD(SLAT)
      SLONG = DMTOD(SLONG)
      XNU = -39.04 + .04*PIXNO1
        CALL CZSUB3(XINCLP,ALTCOR,XLAT,XLONG,TILT,XNU,XLATO,
     * XLONGO,PITCH,ROLL,YAW,IERR)
        CALL CZSUB4 (XINCLP,PERIOD,XLATO,XLONGO,XLONGP,T,IERR)
      T1 = T
        CALL CZSUB5(XINCLP,ALTCOR,PERIOD,XLONGP,SLAT,SLONG,TILT,
     * XLATO,XLONGO,T,XNU,PITCH,ROLL,YAW,IERR)
      PIXNO2 = (XNU + 39.04)/.04
      SCANO2 = SCANO1 - (T - T1)/0.12375
C
      XLAT = DTODM(XLAT)
      XLONG = DTODM(XLONG)
      SLAT = DTODM(SLAT)
      SLONG = DTODM(SLONG)
C
      WRITE (TTY,4000)
      WRITE (TTY,4011) XLAT,XLONG,PIXNO1,SCANO1
      WRITE (TTY,4053) SLAT,SLONG
      WRITE (TTY,4013) TILT
      WRITE (TTY,4001)
      IF (INTL(XNU).EQ.-9998) THEN
            WRITE (TTY,4052)
         ELSE
            WRITE (TTY,4055) PIXNO2,SCANO2
         END IF
      WRITE (TTY,4002)
C
      WRITE (PTR,4000)
      WRITE (PTR,4011) XLAT,XLONG,PIXNO1,SCANO1
      WRITE (PTR,4053) SLAT,SLONG
      WRITE (PTR,4013) TILT
      WRITE (PTR,4001)
      IF (INTL(XNU).EQ.-9998) THEN
            WRITE (PTR,4052)
         ELSE
            WRITE (PTR,4055) PIXNO2,SCANO2
         END IF
      WRITE (PTR,4002)
C
C
      WRITE (TTY,*) 'Use RETURN key to continue program when ready.'
      READ  (TTY,5000)
      GO TO 10
C
C              DegMin-DecDeg and Time-Scan Lines conversions
C
   80 CONTINUE
 3080 FORMAT ('Enter value to be converted, space, and identifier:')
```

```
 3085  FORMAT (T15,'" L"  = Scan lines',/,T15,'" S"  = Seconds',
     *  /,T15,'" DM" = Degrees Minutes',/,T15,'" DD" = Decimal Degrees',
     *   /,T15,' "Q"  = QUIT')
 3083  FORMAT (F7.1,' seconds equals',I5,' scan lines.')
 3084  FORMAT (I5,' scan lines equals',F7.1,' seconds.')
 3081  FORMAT (F10.4,' degrees.minutes equals',F10.4,' decimal degrees')
 3082  FORMAT (F10.4,' decimal degrees equals',F10.4,' degrees.minutes')
       IFIRST = 1
   81  CONTINUE
       WRITE (TTY,3080)
       IF (IFIRST.EQ.1) WRITE (TTY,3085)
       READ  (TTY,1040) CHAR
       IF (INDEX(CHAR,'Q').NE.0) GO TO 10
       READ  (CHAR,*) VAL
       IF (INDEX(CHAR,'M').NE.0) THEN
             WRITE (TTY,3081) VAL ,DMTOD(VAL)
          ELSE IF (INDEX(CHAR,'S').NE.0) THEN
             WRITE (TTY,3083) VAL,VAL/0.12375
          ELSE IF (INDEX(CHAR,'L').NE.0) THEN
             WRITE (TTY,3084) VAL,VAL*0.12375
          ELSE
             WRITE (TTY,3082) VAL,DTODM(VAL)
          END IF
       WRITE (TTY,4002)
       IFIRST = IFIRST + 1
       GO TO 81
C
C
C                              Change Roll, Pitch, Yaw values
C
   90  CONTINUE
 3090  FORMAT ('Enter new Roll, Pitch, Yaw values.  DECIMAL DEGREES!')
       WRITE (TTY,3090)
       READ  (TTY,*) ROLL,PITCH,YAW
       WRITE (TTY,4090) ROLL,PITCH,YAW
       WRITE (PTR,4090) ROLL,PITCH,YAW
       WRITE (PTR,4002)
C
       GO TO 10
C
   99  CONTINUE
       WRITE (TTY,*) 'ERROR ON FILE OPENING'
       STOP 'EXIT CZCS Navigation'
C
  100  CONTINUE
       ENDFILE (PTR)
       CLOSE (PTR)
       STOP 'EXIT CZCS Navigation'
       END
```

```
      REAL FUNCTION DMTOD*8(X)
        REAL*8 X,P3,P2,DINT,DMOD,DSIGN,DABS
      REAL*8 DMTOD
C
        P3 = DABS(X)
        P2 = DINT(P3) + DMOD(P3,1.0D0)*100.0/60.0
        DMTOD = P2*DSIGN(1.0D0,X)
        RETURN
        END
      REAL FUNCTION DTODM*8(X)
         IMPLICIT REAL*8(A-H,O-Z)
        REAL*8 P3,P2,DINT,DMOD,X,DSIGN,DABS
        P3= DABS(X) + .0000005
        P2 = DINT(P3) + DMOD(P3,1.0D0)*60.0/100.0
        DTODM = P2*DSIGN(1.0D0,X)
        RETURN
        END
```