National Mapping Program
Technical Instructions

GCTP
General Cartographic Transformation Package

Software Documentation

U.S. Department of the Interior
U.S. Geological Survey
National Mapping Division

CONTENTS

LIST OF PAGES

A complete and current copy of the Software Documentation for the General Cartographic Transformation Package consists of the pages (and most recent creation or revision dates) listed below.

Page	Date	Page	Date	Page	Date
ii	09/90	34	09/90	72	09/90
iii	09/90	35	09/90	73	09/90
iv	09/90	36	09/90	74	09/90
v	10/94	37	09/90	75	09/90
1	09/90	38	09/90	76	09/90
2	09/90	39	09/90	77	09/90
3	12/91	40	09/90	78	09/90
4	08/94	41	09/90	79	09/90
5	09/90	42	09/90	80	09/90
6	12/91	43	09/90	81	09/90
6a	12/91	44	09/90	82	09/90
7	09/90	45	09/90	83	09/90
8	09/90	46	09/90	84	09/90
9	10/94	47	09/90	85	09/90
10	09/90	48	09/90	86	09/90
11	09/90	49	09/90	87	09/90
12	09/90	50	09/90		
13	09/90	51	09/90		
14	09/90	52	09/90		
15	09/90	53	09/90		
16	09/90	54	09/90		
17	09/90	55	09/90		
18	09/90	56	09/90		
19	09/90	57	09/90		
20	09/90	58	09/90		
21	09/90	59	09/90		
22	12/91	60	09/90		
23	09/90	61	09/90		
24	09/90	62	09/90		
25	09/90	63	09/90		
26	09/90	64	09/90		
27	09/90	65	09/90		
28	09/90	66	09/90		
29	09/90	67	09/90		
30	09/90	68	09/90		
31	08/94	69	08/94		
32	09/90	70	08/94		
33	09/90	71	08/94		

Software Documentation for the General Cartographic Transformation Package

1. GENERAL INFORMATION

1.1 SUMMARY

The General Cartographic Transformation Package (GCTP) is a system of computer subroutines, written in FORTRAN, designed to permit the transformation of coordinate pairs from one map projection to another. GCTP should not be used to transform coordinates between spheroids, because a datum shift should be applied to the geographic coordinates in most cases. It is a subroutine package that must be linked to and called by other FORTRAN programs. The GCTP is the standard computer software used by the National Mapping Division for map projection computations. The mathematical algorithms used in GCTP meet the accuracy specifications of USGS Circular 878-B, "Representation of Geographic Point Locations for Information Interchange," which has been proposed as Federal Information Processing Standards Publication 70-1. This software is approved for use with all products of the National Mapping Program.

1.2 ENVIRONMENT

The GCTP subroutines are generally used as object modules, which are linked to application programs. The software is presently used in the Digital Line Graph Production System (PROSYS), the Cartographic Automated Mapping (USGS CAM1 or K971) system, L176 Batch General Map Transformation, L177 Interactive General Map Transformation, J898 General Map Transformation Driver, and the Universal Projection Plotting System (UPPS).

1.3 REFERENCES

Snyder, J.P., and Voxland, P.M., 1989, An album of map projections: U.S. Geological Survey Professional Paper 1453, 249 p.

Snyder, J.P., 1987, Map projections - A working manual: U.S. Geological Survey Professional Paper 1395, 383 p.

Snyder, J.P., 1982, Map projections used by the U.S. Geological Survey: U.S. Geological Survey Bulletin 1532, 313 p.

Claire, C.N., 1968, State plane coordinates by automatic data processing: Coast and Geodetic Survey Publication No. 62-4, 68 p.

Specifications for representation of geographic point locations for information interchange, 1983, U.S. Geological Survey Circular 878-B, 24 p.

Universal Transverse Mercator grid, 1958, U.S. Department of the Army Technical Report TR 5-241-8, 64 p.

2. PACKAGE DESCRIPTION

2.1 SYSTEM DESCRIPTION

The package allows conversion of a coordinate pair in one projection to another projection, or linear or angular unit conversion of a coordinate pair within one system. Appendix A provides information for State Plane Coordinate System zone codes, appendix B for Universal Transverse Mercator zone coverages, and appendix C for descriptions of the parameters necessary for the projections.

2.1.1 Program and Solution Method

The mathematical algorithms employed are listed in USGS Professional Paper 1395, except for the Robinson projection which is documented in USGS Professional Paper 1453 (see references, section 1.3).

2.1.2 Input

All input to the package is handled through subroutine GTPZ0. Input includes coordinates, projection system code, zone code, projection parameters, units code, and spheroid code, and desired output projection code, zone code, projection parameters, and units code.

2.1.3 Processing

The individual subroutines and functions are described in sections 2.2 through 2.51.

2.1.4 Output

All output for error-free computations is passed through variables in the call line for subroutine GTPZ0. The output is the desired coordinates. When errors occur, a non-zero code is returned to the calling routine and an error message is output to the logical unit specified for variable LEMSG.

2.1.5 Interfaces

The package is included in the application program at link-edit time into the load module (on the Amdahl) or the executable module at link time on other computers. The office listed below should be contacted for the proper data set name and linkage procedures.

U.S. Geological Survey
National Mapping Division
Office of Production Operations
Branch of Operations Policy
National Center Mail Stop 510
Reston, Virginia 22092

2.1.6 Run Description

The primary access to the package is through a call to subroutine GTPZ0. Section 2.2 describes the required variables to be passed to the routine and to be received back to the calling program. The rest of the subprograms for the package are described in sections 2.3 through 2.51.

The standard FORTRAN naming conventions are followed in this documentation. All real or floating-point variables are REAL*4 or REAL*8. All integers are INTEGER*2 or INTEGER*4.

2.2 SUBROUTINE GTPZ0

2.2.1 Summary

Subroutine GTPZ0 is the primary access routine to the GCTP. "Driver" or main programs would normally call the package through this routine. The use of constants in the call line is not recommended as some of the variables are changed.

2.2.2 Call Line

The call is as follows:

```
 CALL GTPZ0(CRDIN,INSYS,INZONE,TPARIN,INUNIT,INSPH,IPR,JPR,LEMSG,
X      LPARM,CRDIO,IOSYS,IOZONE,TPARIO,IOUNIT,LN27,LN83,FN27,FN83,
X      LENGTH,IFLG)
```

(1) CRDIN is the REAL*8 array of two input coordinates (X-Y, longitude-latitude, etc.). The nature of the coordinates is defined by INSYS, INZONE, and INUNIT. The east-west dimension (X, longitude, easting) is first, followed by the north-south (Y, latitude, northing)

Sign conventions: Latitude--North is plus, south is minus
Longitude--East is plus, west is minus

(2) INSYS is the INTEGER*4 input projection system code:

0 = Geographic (default)	1 = Universal Transverse Mercator
2 = State Plane Coordinates	3 = Albers Conical Equal-Area
4 = Lambert Conformal Conic	5 = Mercator
6 = Polar Stereographic	7 = Polyconic
8 = Equidistant Conic	9 = Transverse Mercator
10 = Stereographic	11 = Lambert Azimuthal Equal-Area
12 = Azimuthal Equidistant	13 = Gnomonic
14 = Orthographic	15 = General Vertical Near-Side Perspective
16 = Sinusoidal	17 = Equirectangular
18 = Miller Cylindrical	19 = Van der Grinten
20 = Oblique Mercator	21 = Robinson
22 = Space Oblique Mercator	23 = Modified Stereographic Conformal (Alaska)

(3) INZONE is the INTEGER*4 input zone code number for the Universal Transverse Mercator (UTM) when INSYS=1, and for the State Plane Coordinates System when INSYS=2. See appendix A for the State Plane zone codes and appendix B for the coverage of the UTM zones. For the UTM in the Southern Hemisphere, INZONE must be negative. When INSYS=0, INZONE is not relevant for Geographic coordinates.

The initialization of the projection will always take place during the first call to the corresponding projection routine. Subsequent calls will result in the recomputation of the initialization parameters if the zone number is non-zero and is different from the zone number used in the preceding call. If the zone number is identical to the preceding call, that previous initialization will be used.

When INSYS is greater than 2, the use of a non-zero INZONE associates that number with the set of parameters that is input. If the INZONE value remains unchanged with subsequent calls, the parameters are reused (the same zone definition is used) without reinitialization of the projection.

(4) TPARIN is a REAL*8 array of 15 input projection parameters. If INSYS is greater than 2, this array must be provided by the calling program. See the individual projection subroutines for the definition of the required items. All longitudes and latitudes in the parameter array are assumed to be in the standard packed DMS format (±DDDMMMSSS.SSS). Function PAKCZ0 must be used to convert the alternate packed DMS format (±DDDMMSS.SSS) to the standard packed DMS format for all angular elements in the parameter array before subroutine GTPZ0 is called. If the eccentricity is zero, a sphere of radius A is assumed.

(5) INUNIT is the INTEGER*4 input units code for the values in array CRDIN:

0 = radians (default)	1 = U.S. feet
2 = meters	3 = seconds of arc
4 = degrees of arc	5 = International ft.
6 = table supplying the unit code, which is legislated for the State zone selected	

(6) INSPH is the INTEGER*4 input-output spheroid code from the following list:

Code	Spheroid	Code	Spheroid
0	Clarke 1866	10	Modified Everest
1	Clarke 1880	11	Modified Airy
2	Bessel	12	WGS 84
3	New International 1967	13	Southeast Asia
4	International 1909	14	Australian National
5	WGS 72	15	Krassovsky
6	Everest	16	Hough
7	WGS 66	17	Mercury 1960
8	GRS 1980	18	Modified Mercury 1968
9	Airy	19	Normal Sphere

If the user wishes to supply constants for a spheroid other than those above, a negative INSPH value must be used, and the semimajor axis and semiminor axis or eccentricity squared must be supplied in TPARIN and TPARIO.

(7) IPR is the INTEGER*4 printout flag for printing error messages. If IPR is zero, error messages will be printed on logical unit LEMSG. If IPR is not zero, error messages will not be printed.

(8) JPR is the INTEGER*4 printout flag for printing projection parameters. If JPR is zero, projection parameters will be printed on logical unit LPARM. If JPR is not zero, projection parameters will not be printed.

(9) LEMSG is the INTEGER*4 logical unit number where error messages will be printed.

(10) LPARM is the INTEGER*4 logical unit number where projection parameters will be printed.

(11) CRDIO is the REAL*8 output <u>array</u> of the transformed coordinates. See CRDIN for explanation.

(12) IOSYS is the INTEGER*4 output projection system code. See INSYS.

(13) IOZONE is the INTEGER*4 output zone number. See INZONE.

(14) TPARIO is the REAL*8 <u>array</u> of 15 parameters for the output projection. This array must be supplied by the calling program if IOSYS is not 1 or 2, or if INSPH is negative. See TPARIN for explanation.

(15) IOUNIT is the INTEGER*4 output units code. See INUNIT.

(16) LN27 is the INTEGER*4 logical unit number of the direct access file where the NAD 1927 State Plane zone parameters are located.

(17) LN83 is the INTEGER*4 logical unit number of the direct access file where the NAD 1983 State Plane zone parameters are located.

(18) FN27 is the CHARACTER*128 file name of the direct access file containing the NAD 1927 State Plane zone parameters. FN27 may be up to 128 characters long.

(19) FN83 is the CHARACTER*128 file name of the direct access file containing the NAD 1983 State Plane zone parameters. FN83 may be up to 128 characters long.

(20) LENGTH is the INTEGER*4 record length of direct access files FN27 and FN83. For the Amdahl, Concurrent, and Gould computers, LENGTH is 108 bytes. For VAX computers, LENGTH is 27 words.

(21) IFLG is the INTEGER*4 error flag after the transformation:

0 = No errors	1 = INSYS is illegal
2 = IOSYS is illegal	3 = INUNIT is illegal
4 = IOUNIT is illegal	5 = INSYS and INUNIT inconsistent
6 = IOSYS and IOUNIT inconsistent	7 = INZONE is illegal
8 = IOZONE is illegal	
11= INUNIT or IOUNIT is illegal	12= INUNIT and IOUNIT are inconsistent

For higher numbers, the error flag has been set by one of the projection subroutines.

2.2.3 Algorithm

(1) Checks the validity of the codes:
INSYS within the range 0 to 23
IOSYS within the range 0 to 23
INUNIT within the range 0 to 6
IOUNIT within the range 0 to 6

(2) Checks the validity of the units by calling UNTFZ0.

(3) Quick return with a units change only if INSYS=IOSYS with INZONE=IOZONE.

(4) If INSYS is geographic (= 0), perform FORWARD computation; otherwise do INVERSE for projection selected (01 through 23).

(5) For each projection, use initialization subroutine PJINIT first, then use subroutine PJXXZ0 to transform the coordinates (XX is 01 through 23).

(6) Return to the calling program.

2.2.4 Error Messages

See IFLG under the call line description (item 21, section 2.2.2). The error flag IFLG is set to 0 if the computation was successful with no errors, or non-zero if there was an error. The calling program must provide the display of the error code, any appropriate message, or other response to the error.

2.2.5 Size

Approximate number of source statements (excluding comments): 257
Approximate size of object code in bytes (Amdahl): 11,224

2.3 FUNCTION ADJLZ0

DOUBLE PRECISION FUNCTION ADJLZ0(LON)

2.3.1 Summary

This function adjusts angle LON, a longitude angle, so the magnitude of the absolute value is less than PI radians (180°). All values are REAL*8.

2.3.2 Algorithm

Two times PI is subtracted (if longitude LON is positive) or added (if LON is negative) from or to LON until the value is not greater than PI radians and less than -PI radians.

2.3.3 Error Messages

None

2.3.4 Size

Approximate number of source statements (excluding comments): 9
Approximate size of object code in bytes (Amdahl): 756

2.4 FUNCTION ASINZ0

DOUBLE PRECISION FUNCTION ASINZ0(CON)

2.4.1 Summary

This function adjusts round-off errors before the arc sine function DASIN is called. Some computers cannot compute the arc sine if the absolute value of the argument is the slightest bit larger than one. All values are REAL*8.

2.4.2 Algorithm

After CON is tested and the absolute of CON is assured to be no greater than one, the arc sine function DASIN is called with the possibly revised value of CON as the argument.

2.4.3 Error Messages

None

2.4.4 Size

Approximate number of source statements (excluding comments): 9

Approximate size of object code in bytes (Amdahl): 712

2.5 FUNCTION DMSPZ0

DOUBLE PRECISION FUNCTION DMSPZ0(SGNA,DEGS,MINS,SECS)

2.5.1 Summary

This function converts an angle in degrees, minutes, and seconds to packed DMS format ±DDDMMMSSS.SSS.

2.5.2 Algorithm

SGNA is the sign that is either blank (for positive) or a minus sign set by NEG, which is initialized by DATA NEG/'-'/, and is stored as CHARACTER*1.

DEGS is the degrees value stored as INTEGER*4.

MINS is the minutes value stored as INTEGER*4.

SECS is the seconds value stored as a REAL*4.

The packed DMS format is built as:

degrees * 1000000 + minutes * 1000 + seconds

Example: +50 degrees, 30 minutes, 36.25 seconds becomes

DMSPZ0 = 50030036.25 stored as a REAL*8 variable.

2.5.3 Error Messages

None

2.5.4 Size

Approximate number of source statements (excluding comments): 12

Approximate size of object code in bytes (Amdahl): 1,004

2.6 FUNCTION E0FNZ0

DOUBLE PRECISION FUNCTION E0FNZ0(ESQ)

2.6.1 Summary

This function computes the constant "e_0" from the eccentricity squared ESQ. "e_0" is used in a series for calculating the distance along a meridian. All variables are REAL*8.

2.6.2 Algorithm

$$e_0 = 1 - \frac{ESQ}{4} \left(1 + \frac{ESQ}{16} \left(3 + \frac{5 * ESQ}{4} \right) \right)$$

2.6.3 Error Messages

None

2.6.4 Size

Approximate number of source statements (excluding comments): 6

Approximate size of object code in bytes (Amdahl): 784

2.7 FUNCTION E1FNZ0

DOUBLE PRECISION FUNCTION E1FNZ0(ESQ)

2.7.1 Summary

This function computes the constant "e_1" from input of the eccentricity squared ESQ. "e_1" is used in a series to calculate a distance along a meridian. All variables are REAL*8.

2.7.2 Algorithm

$$e_1 = \frac{3 * ESQ}{8} \left(1 + \frac{ESQ}{4} \left(1 + \frac{15 * ESQ}{32} \right) \right)$$

2.7.3 Error Messages

None

2.7.4 Size

Approximate number of source statements (excluding comments): 7

Approximate size of object code in bytes (Amdahl): 752

2.8 FUNCTION E2FNZ0

DOUBLE PRECISION FUNCTION E2FNZ0(ESQ)

2.8.1 Summary

This function computes the constant "e_2" from input of the eccentricity squared ESQ. "e_2" is used in a series to calculate a distance along a meridian. All variables are REAL*8.

2.8.2 Algorithm

$$e_2 = \frac{15}{256} * ESQ^2 * \left(1 + \frac{3 * ESQ}{4} \right)$$

2.8.3 Error Messages

None

2.8.4 Size

Approximate number of source statements (excluding comments): 7

Approximate size of object code in bytes (Amdahl): 700

2.9 FUNCTION E3FNZ0

DOUBLE PRECISION FUNCTION E3FNZ0(ESQ)

2.9.1 Summary

This function computes the constant "e_3" from input of the eccentricity squared ESQ. "e_3" is used in a series to calculate a distance along a meridian. All variables are REAL*8.

2.9.3 Algorithm

$$e_3 = ESQ^3 * \left(\frac{35}{3072} \right)$$

2.9.3 Error Messages

None

2.9.4 Size

Approximate number of source statements (excluding comments): 5

Approximate size of object code in bytes (Amdahl): 596

2.10 FUNCTION E4FNZ0

DOUBLE PRECISION FUNCTION E4FNZ0(ECC)

2.10.1 Summary

This function computes constant "e_4" from input of the eccentricity of the spheroid ECC. This constant is used in the Polar Stereographic projection. All variables are REAL*8.

2.10.2 Algorithm

e_4 = SQRT [$(1 + ECC)^{1+ECC}$ * $(1 - ECC)^{1-ECC}$]

2.10.3 Error Messages

None

2.10.4 Size

Approximate number of source statements (excluding comments): 8

Approximate size of object code in bytes (Amdahl): 776

2.11 FUNCTION MLFNZ0

DOUBLE PRECISION FUNCTION MLFNZ0(E0,E1,E2,E3,PHI)

2.11.1 Summary

This function computes the value of "M," which is the distance along a meridian from the Equator to latitude PHI. All variables are REAL*8. PHI is the latitude; E0, E1, E2, and E3 are constants as computed by functions E0FNZ0, E1FNZ0, E2FNZ0, and E3FNZ0, respectively.

2.11.2 Algorithm

$$M = e_0 * PHI - e_1 * SIN(2 * PHI) + e_2 * SIN(4 * PHI) - e_3 * SIN(6 * PHI)$$

2.11.3 Error Messages

None

2.11.4 Size

Approximate number of source statements (excluding comments): 6

Approximate size of object code in bytes (Amdahl): 952

2.12 FUNCTION MSFNZ0

DOUBLE PRECISION FUNCTION MSFNZ0(ECC,SINPHI,COSPHI)

2.12.1 Summary

This function computes the constant "m," which is the radius of a parallel of latitude PHI divided by the semimajor axis. All variables are REAL*8.

2.12.2 Algorithm

$$m = \frac{COS(PHI)}{(\ 1 - ECC^2 * SIN^2(PHI))^{1/2}}$$

2.12.3 Error Messages

None

2.12.4 Size

Approximate number of source statements (excluding comments): 7

Approximate size of object code in bytes (Amdahl): 768

2.13 FUNCTION PAKCZ0

DOUBLE PRECISION FUNCTION PAKCZ0(PAK)

2.13.1 Summary

This function converts an angle PAK in alternate packed DMS format ±DDDMMSS.SSS to standard packed DMS format ±DDDMMMSSS.SSS. PAK is REAL*8.

2.13.2 Algorithm

Angle PAK is portioned into four variables: sign SGNA, degrees DEGS, minutes MINS, and seconds SECS in the same manner as function PAKDZ0.

SGNA is stored as CHARACTER*1.
DEGS is stored as INTEGER*4.
MINS is stored as INTEGER*4.
SECS is stored as REAL*8.

The output angle PAKCZ0 in standard packed DMS format is:

degrees * 1000000 + minutes * 1000 + seconds

```
Example:   PAK = 503036.25 yields
           SGNA = ' '
           DEGS = 50
           MINS = 30
           SECS = 36.25
           PAKCZ0 = 50030036.25
```

2.13.3 Error Messages

None

2.13.4 Size

Approximate number of source statements (excluding comments): 19

Approximate size of object code in bytes (Amdahl): 1,284

2.14 SUBROUTINE PAKDZ0

SUBROUTINE PAKDZ0(PAK,SGNA,DEGS,MINS,SECS)

2.14.1 Summary

This subroutine converts an angle PAK in standard packed DMS format to degrees, minutes, and seconds. PAK is REAL*8.

2.14.2 Algorithm

Angle PAK is portioned into sign, degrees, minutes, and seconds as follows:

- SGNA is the sign as either blank (for positive) or a minus sign (for negative), stored as CHARACTER*1.
- DEGS is the degrees stored as INTEGER*4.
- MINS is the minutes stored as INTEGER*4.
- SECS is the seconds stored as a REAL*4.

The standard packed DMS format is:

degrees * 1000000 + minutes * 1000 + seconds

```
Example:   PAK = 50030036.25 yields
           SGNA = ' '
           DEGS = 50
           MINS = 30
           SECS = 36.25
```

2.14.3 Error Messages

None

2.14.4 Size

Approximate number of source statements (excluding comments): 16

Approximate size of object code in bytes (Amdahl): 1,140

2.15 FUNCTION PAKRZ0

DOUBLE PRECISION FUNCTION PAKRZ0(ANG)

2.15.1 Summary

This function converts a packed DMS angle ANG to radians. All variables are REAL*8.

2.15.2 Algorithm

Function PAKSZ0 is called to convert ANG from packed DMS to seconds of arc.

PAKRZ0 = ANG * 0.484813681095359 * 10^{-5} converts seconds to radians.

2.15.3 Error Messages

None

2.15.4 Size

Approximate number of source statements (excluding comments): 7

Approximate size of object code in bytes (Amdahl): 680

2.16 FUNCTION PAKSZ0

DOUBLE PRECISION FUNCTION PAKSZ0(ANG)

2.16.1 Summary

This function converts a packed DMS angle ANG to seconds. All variables are REAL*8. See sections 2.5 and 2.14 for definition of packed DMS.

2.16.2 Algorithm

(1) The absolute value of the angle is used.

(2) The degrees are separated out:

DEGS = ANG/1000000 (fractional portion truncated)

(3) The minutes are separated out:

MINS = (ANG - DEGS * 1000000) / 1000

(fractional portion truncated)

(4) The seconds are then computed:

SECS = ANG - DEGS * 1000000 - MINS * 1000

(5) The total angle in seconds is computed:

PAKSZ0 = DEGS * 3600.0 + MINS * 60.0 + SECS

(6) The sign is of PAKSZ0 set to that of the input angle.

2.16.3 Error Messages

"ILLEGAL DMS FIELD = . . ." is printed if DEGS exceed 360, MINS exceed 60, or SECS exceed 60.

2.16.4 Size

Approximate number of source statements (excluding comments): 31

Approximate size of object code in bytes (Amdahl): 1,576

2.17 FUNCTION PHI1Z0

DOUBLE PRECISION FUNCTION PHI1Z0(ECC,QS)

2.17.1 Summary

Through an iterative procedure, this function computes the latitude angle PHI1. PHI1 is the equivalent of the latitude PHI for the inverse of the Albers Conical Equal-Area projection. QS is the input angle in radians as computed by QSFNZ0. ECC is the eccentricity. All values are REAL*8 and all angular values are in radians.

2.17.2 Algorithm

(1) The starting value is set: PHI = SIN^{-1}(QS/2)

(2) If ECC is less than 10^{-7}, the starting value is returned.

(3) ESQ = ECC * ECC to compute the eccentricity squared.

(4) DPHI is computed:

$$DPHI = \frac{(1 - ESQ * SIN^2(PHI))^2}{2 * COS(PHI)} * \left[\frac{QS}{1 - ESQ} - \frac{SIN(PHI)}{(1 - ESQ*SIN^2(PHI))} + \frac{1}{2 * ECC} * \ln\left(\frac{(1 - ECC * SIN(PHI))}{(1 + ECC * SIN(PHI))} \right) \right]$$

(5) PHI = PHI + DPHI

(6) If DPHI is not less then 10^{-10}, steps 4 and 5 are repeated up to 15 times.

2.17.3 Error Messages

LATITUDE FAILED TO CONVERGE AFTER "n" ITERATIONS

ECCENTRICITY = . . . QS = . . .

2.17.4 Size

Approximate number of source statements (excluding comments): 28

Approximate size of object code in bytes (Amdahl): 1,840

2.18 FUNCTION PHI2Z0

DOUBLE PRECISION FUNCTION PHI2Z0(ECC,TS)

2.18.1 Summary

The latitude PHI2 is computed using an iterative procedure. PHI2 is PHI for the inverse of the Lambert Conformal Conic and Polar Stereographic projections. ECC is the spheroid eccentricity; TS is the constant "t" as computed by TSFNZ0. All real variables are REAL*8.

2.18.2 Algorithm

(1) A starting PHI is computed:

$$PHI = \frac{PI}{2} - 2 * TAN^{-1}(TS)$$

(2) $$DPHI = \frac{PI}{2} - 2 * TAN^{-1}\left[TS * \left(\frac{(1 - ECC * SIN(PHI))}{(1 + ECC * SIN(PHI))}\right)^{ECC/2}\right] - PHI$$

(3) PHI = PHI + DPHI

(4) If DPHI is not less than 10^{-10}, repeat steps 2 and 3 up to 15 times.

2.18.3 Error Messages

LATITUDE FAILED TO CONVERGE AFTER "n" ITERATIONS

ECCENTRICITY = . . .   TS = . . .

2.18.4 Size

Approximate number of source statements (excluding comments): 25

Approximate size of object code in bytes (Amdahl): 1,660

2.19 FUNCTION PHI3Z0

DOUBLE PRECISION FUNCTION PHI3Z0(ML,E0,E1,E2,E3)

2.19.1 Summary

This function computes PHI3 using an iterative process. PHI3 is the latitude PHI for the inverse of the Equidistant Conic projection. All variables are REAL*8.

ML = Constant computed by MLFNZ0.

E0 = Constant computed by E0FNZ0.

E1 = Constant computed by E1FNZ0.

E2 = Constant computed by E2FNZ0.

E3 = Constant computed by E3FNZ0.

2.19.2 Algorithm

(1) The starting PHI is set to equal ML.

(2) $$DPHI = \frac{ML+E1+SIN(2*PHI)-E2*SIN(4*PHI)+E3*SIN(6*PHI)}{E0} - PHI$$

(3) PHI = PHI + DPHI

(4) If DPHI is not less than 10^{-10}, repeat steps 2 and 3 up to 15 times.

2.19.3 Error Messages

LATITUDE FAILED TO CONVERGE AFTER "n" ITERATIONS

ML = . . .   E0 = . . .

E1 = . . .   E2 = . . .   E3 = . . .

2.19.4 Size

Approximate number of source statements (excluding comments): 21

Approximate size of object code in bytes (Amdahl): 1,652

2.20 SUBROUTINE PHI4Z0

SUBROUTINE PHI4Z0(ESQ,E0,E1,E2,E3,A,B,C,PHI)

2.20.1 Summary

Through an iterative process, this subroutine computes the latitude PHI for the inverse of the Polyconic projection. All real variables are REAL*8.

- ESQ = The spheroid eccentricity squared.
- E0 = From E0FNZ0
- E1 = From E1FNZ0
- E2 = From E2FNZ0
- E3 = From E3FNZ0
- A = Constant transmitted to the function.
- B = Constant transmitted to the function.
- C = Constant developed in the function and transmitted back to the calling routine.

2.20.2 Algorithm

(1) The starting value is set PHI = A

(2) $C = TAN(PHI) * (1 - ESQ * SIN^2(PHI))^{1/2}$

(3) $ML = E0 * PHI - E1 * SIN(2 * PHI) + E2 * SIN(4 * PHI) - E3 * SIN(6 * PHI)$

(4) $MLP = E0 - 2 * E1 * COS(2 * PHI) + 4 * E2 * COS(4 * PHI) - 6 * E3 * COS(6 * PHI)$

(5) $CON1 = 2 * ML + C * (ML^2 + B) - 2 * A * (C * ML + 1)$

(6) $$CON2 = \frac{ESQ * SIN(2 * PHI) * (ML^2 + B - 2 * A * ML)}{2 * C}$$

(7) $CON3 = 2 * (A - ML) * (C * MLP - 2 / (SIN(2 * PHI)) - 2 * MLP$

(8) DPHI = CON1 / (CON2 + CON3)

(9) PHI = PHI + DPHI

(10) If DPHI is not less than 10^{-10}, then repeat steps 2 through 9 up to 15 times.

2.20.3 Error Messages

LATITUDE FAILED TO CONVERGE AFTER "n" ITERATIONS

E0 = . . .   E1 = . . .

E2 = . . .   E3 = . . .

A = . . .   B = . . .

C = . . .

ECCENTRICITY SQUARE = . . .

2.20.4 Size

Approximate number of source statements (excluding comments): 29

Approximate size of object code in bytes (Amdahl): 2,700

2.21 INTRODUCTION TO PROJECTION SUBROUTINES

2.21.1 Summary

Each projection is initialized by subroutine PJINIT. A projection need not be initialized again unless one or more of its parameters change. Each projection is computed by a separate subroutine. Each routine contains two sections: forward (geographic to grid) and inverse (grid to geographic). The characters XX shown below are the projection number cited in appendix C; for example, projection 14, the Orthographic, has the subroutine name of PJ14Z0. For the mathematical formulations not found here, see USGS Professional Papers 1395 and 1453.

2.21.2 Subroutine PJXXZ0 Description

SUBROUTINE PJXXZ0(COORD,CRDIO,INDIC)

This is the generalized subroutine name. COORD is the two-element REAL*8 array containing the input coordinates. CRDIN is the two-element REAL*8 array of output coordinates. INDIC is an INTEGER*2 indicator, which must be either zero to specify a forward computation or one to specify an inverse computation.

2.21.3 Error Messages

The error messages are described in the individual subroutines. PROJECTION WAS NOT INITIALIZED will be generated if parameters are missing when a forward or inverse is called before initialization.

2.21.4 COMMON Storage

The COMMON block PRINZ0 contains four parameters defining whether printout is to occur. If the first INTEGER*4 word IPEMSG is zero, error messages will print. Printing of error messages will be suppressed if IPEMSG is non-zero. The second INTEGER*4 word IPELUN is the logical unit where the error messages will print. If the third INTEGER*4 word IPPARM is zero, the initialization parameters will print. Printing of initialization parameters will be suppressed if IPPARM is non-zero. The fourth INTEGER*4 word IPPLUN is the logical unit where the projection initialization parameters will print.

The COMMON block ERRMZ0 contains one INTEGER*4 word IERROR consisting of the error code returned by the various routines.

The COMMON block ELLPZ0 contains the REAL*8 spheroid values AZ, EZ, ESZ, E0Z, E1Z, E2Z, E3Z, and E4Z.

The COMMON block SPHRZ0 has the REAL*8 reference spheroid radius AZZ.

The COMMON block PROJZ0 contains an INTEGER*4 code number of the previous input projection IPROJ, which is passed only between subroutine GTPZ0 and subroutine SPHDZ0.

The COMMON block SPCS contains five INTEGER*4 values and two CHARACTER*128 values, which are passed only between subroutine GTPZ0 and subroutine PJINIT. ISPHER is the spheroid code number. LU27 is the logical unit number for NAD 1927 State Plane zone constants file FILE27. LU83 is the logical unit number of the NAD 1983 State Plane zone constants file FILE83. LEN is the length of each direct access file record in bytes or words (see section 2.2.2). MSYS is the projection code (4, 7, 9, or 20) of the current State Plane Coordinate zone.

The COMMON block TOGGLE contains an array SWITCH of 23 INTEGER*4 values, which indicate the initialization status of each of the 23 projections. A SWITCH value of zero indicates initialization has not been performed. A non-zero value indicates initialization has been performed.

The COMMON block NORM contains nine REAL*8 constants for the Space Oblique Mercator projection, which are only passed between subroutine PJ22Z0 and subroutine SERAZ0.

The COMMON block PJXX (XX from 02 to 23) contains REAL*8 parameters passed between subroutine PJINIT and subroutine PJXXZ0, respectively.

2.22 SUBROUTINE PJINIT

SUBROUTINE PJINIT(ISYS,ZONE,DATA)

2.22.1 Summary

Subroutine PJINIT is used for initialization of any one of the 23 projections in GCTP. ISYS is the INTEGER*4 code number of the projection and must be from 0 to 23 as described in section 2.2.2. Zone is the INTEGER*4 zone number and must be non-zero for any projection. For ISYS = 1 (UTM), ZONE must be from -60 to +60. When ZONE = 0, the optimum UTM zone will be computed. If the user knows the UTM zone number needed, it is wiser to use it especially at zone boundaries because of the ambiguity there, rather than let the program compute the optimum UTM zone. For ISYS = 2 (State Plane), ZONE must be one of the zone codes from appendix A. State Plane coordinates can only be computed when the Clarke 1866 spheroid (INSPH = 0) is used for the North American Datum of 1927 (NAD 1927), or when the Global Reference System of 1980 (GRS 1980) spheroid (INSPH = 8) is used for the North American Datum of 1983 (NAD 1983). DATA is the 15-element REAL*8 parameter array described in the following sections for each projection and in appendix C.

2.22.3 Error Messages

Code	Projection	Message
011	PJ01Z0	ILLEGAL ZONE NO.: . . .
020	PJ02Z0	SPHEROID NO. . . . IS INVALID FOR STATE PLANE TRANSFORMATIONS
021	PJ02Z0	ILLEGAL ZONE NO.: . . . FOR SPHEROID NO.: . . .
031	PJ03Z0	EQUAL LATITUDES FOR STANDARD PARALLELS ON OPPOSITE SIDES OF THE EQUATOR
041	PJ04Z0	EQUAL LATITUDES FOR STANDARD PARALLELS ON OPPOSITE SIDES OF EQUATOR

Code	Projection	Message
081	PJ08Z0	EQUAL LATITUDES FOR STANDARD PARALLELS ON OPPOSITE SIDES OF EQUATOR
201	PJ20Z0	INPUT DATA ERROR (Format A)
202	PJ20Z0	INPUT DATA ERROR (Format B)
221	PJ22Z0	LANDSAT NUMBER . . . AND/OR PATH NUMBER . . . ARE OUT OF RANGE

2.22.4 Size

Approximate number of source statements (excluding comments): 969

Approximate size of object code in bytes (Amdahl): 48,900

2.22.5 Restrictions

None

2.23 SUBROUTINE PJ01Z0

Projection: Universal Transverse Mercator

2.23.1 Definition of Parameter Array

1 Longitude of any point in the zone.
2 Latitude of any point in the zone.
  NOTE: The above are required only for a forward computation when the zone number is zero.
3 Not used
4 Not used
5 Not used
6 Not used
7 Not used
8 Not used
9 Not used
10 Not used
11 Not used
12 Not used
13 Not used
14 Temporary storage of element 1 above
15 Temporary storage of element 2 above

2.23.2 Error Messages

013 Forward: PROJECTION WAS NOT INITIALIZED

014 Inverse: PROJECTION WAS NOT INITIALIZED

2.23.3 Size

Approximate number of source statements (excluding comments): 38

Approximate size of object code in bytes (Amdahl): 1,480

2.23.4 Restrictions

Requires PJ09Z0, "Transverse Mercator."

2.24 SUBROUTINE PJ02Z0

Projection: State Plane Coordinate Systems

2.24.1 Definition of Parameter Array

1 Not used
2 Not used
3 Used for Transverse Mercator, Lambert, and Oblique Mercator (see appendix C)
4 Used for Lambert and Oblique Mercator (see appendix C)
5 Used for Transverse Mercator, Lambert, Polyconic*, and Oblique Mercator (see appendix C)
6 Used for Transverse Mercator, Lambert, Polyconic*, and Oblique Mercator (see appendix C)
7 Used for Transverse Mercator, Lambert, Polyconic*, and Oblique Mercator (see appendix C)
8 Used for Transverse Mercator, Lambert, Polyconic*, and Oblique Mercator (see appendix C)
9 Not used
10 Not used
11 Not used
12 Not used
13 Used for Oblique Mercator (see appendix C)
14 Not used
15 Not used

* The Polyconic projection is used as an approximation to the Azimuthal Equidistant projection for the Guam zone.

2.24.2 Error Messages

023 Forward: PROJECTION WAS NOT INITIALIZED

024 Forward: FAILED TO CONVERGE

025 Inverse: PROJECTION WAS NOT INITIALIZED

026 Inverse: FAILED TO CONVERGE

XXX Other messages may be provided by individual projection routines

2.24.3 Size

Approximate number of source statements (excluding comments): 63

Approximate size of object code in bytes (Amdahl): 1,860

2.24.4 Restrictions

Requires PJ04Z0, PJ07Z0, PJ09Z0, or PJ20Z0, depending upon which projection the specified State zone requires.

2.25 SUBROUTINE PJ03Z0

Projection: Albers Conical Equal-Area

2.25.1 Definition of Parameter Array

1 Semimajor axis of spheroid
2 Eccentricity squared OR semiminor axis of spheroid
3 Latitude of first standard parallel
4 Latitude of second standard parallel
5 Longitude of central meridian
6 Latitude of origin of projection
7 False easting at central meridian
8 False northing at origin
9 Not used
10 Not used
11 Not used
12 Not used
13 Not used
14 Not used
15 Not used

2.25.2 Error Messages

033 Forward: PROJECTION WAS NOT INITIALIZED

034 Inverse: PROJECTION WAS NOT INITIALIZED

2.25.3 Size

Approximate number of source statements (excluding comments): 64

Approximate size of object code in bytes (Amdahl): 2,904

2.25.4 Restrictions

None

2.26 SUBROUTINE PJ04Z0

Projection: Lambert Conformal Conic

2.26.1 Definition of Parameter Array

1 Semimajor axis of spheroid
2 Eccentricity squared OR semiminor axis of spheroid
3 Latitude of first standard parallel
4 Latitude of second standard parallel
5 Longitude of central meridian
6 Latitude of origin of projection
7 False easting at central meridian
8 False northing at origin
9 Not used
10 Not used
11 Not used
12 Not used
13 Not used
14 Not used
15 Not used

2.26.2 Error Messages

043 Forward: PROJECTION WAS NOT INITIALIZED

044 Forward: POINT CANNOT BE PROJECTED

045 Inverse: PROJECTION WAS NOT INITIALIZED

2.26.3 Size

Approximate number of source statements (excluding comments): 71

Approximate size of object code in bytes (Amdahl): 2,924

2.26.4 Restrictions

None

2.27 SUBROUTINE PJ05Z0

Projection: Mercator

2.27.1 Definition of Parameter Array

1 Semimajor axis of spheroid
2 Eccentricity squared OR semiminor axis of spheroid
3 Not used
4 Not used
5 Longitude of central meridian
6 Latitude of true scale
7 False easting at central meridian
8 False northing applied to all coordinates
9 Not used
10 Not used
11 Not used
12 Not used
13 Not used
14 Not used
15 Not used

2.27.2 Error Messages

052 Forward: PROJECTION WAS NOT INITIALIZED

053 Forward: TRANSFORMATION CANNOT BE COMPUTED AT THE POLES

054 Inverse: PROJECTION WAS NOT INITIALIZED

2.27.3 Size

Approximate number of source statements (excluding comments): 55

Approximate size of object code in bytes (Amdahl): 2,388

2.27.4 Restrictions

None

2.28 SUBROUTINE PJ06Z0

Projection: Polar Stereographic

2.28.1 Definition of Parameter Array

1 Semimajor axis of spheroid
2 Eccentricity squared OR semiminor axis of spheroid
3 Not used
4 Not used
5 Longitude directed straight down below pole of map
6 Latitude of true scale
7 False easting applied to all coordinates
8 False northing applied to all coordinates
9 Not used
10 Not used
11 Not used
12 Not used
13 Not used
14 Not used
15 Not used

2.28.2 Error Messages

062 Forward: PROJECTION WAS NOT INITIALIZED

063 Inverse: PROJECTION WAS NOT INITIALIZED

2.28.3 Size

Approximate number of source statements (excluding comments): 64

Approximate size of object code in bytes (Amdahl): 2,712

2.28.4 Restrictions

None

2.29 SUBROUTINE PJ07Z0

Projection: Polyconic

2.29.1 Definition of Parameter Array

1 Semimajor axis of spheroid
2 Eccentricity squared OR semiminor axis of spheroid
3 Not used
4 Not used
5 Longitude at central meridian
6 Latitude of origin of projection
7 False easting applied to all coordinates
8 False northing applied to all coordinates
9 Not used
10 Not used
11 Not used
12 Not used
13 Not used
14 Not used
15 Not used

2.29.2 Error Messages

072 Forward: PROJECTION WAS NOT INITIALIZED

073 Inverse: PROJECTION WAS NOT INITIALIZED

2.29.3 Size

Approximate number of source statements (excluding comments): 67

Approximate size of object code in bytes (Amdahl): 2,884

2.29.4 Restrictions

The inverse computation will not converge if the longitude is greater than 90° from central meridian.

2.30 SUBROUTINE PJ08Z0

Projection: Equidistant Conic

2.30.1 Definition of Parameter Array

	Format A (one standard parallel)		Format B (two standard parallels)
1	Semimajor axis of spheroid	OR	same
2	Eccentricity squared or semiminor axis of spheroid	OR	same
3	Latitude of standard parallel	OR	latitude first parallel
4	Not used	OR	latitude second parallel
5	Longitude at central meridian	OR	same
6	Latitude projection origin	OR	same
7	False easting applied to all coordinates	OR	same
8	False northing applied to all coordinates	OR	same
9	Zero	OR	non-zero
10	Not used		
11	Not used		
12	Not used		
13	Not used		
14	Not used		
15	Not used		

2.30.2 Error Messages

083 Forward: PROJECTION WAS NOT INITIALIZED

084 Inverse: PROJECTION WAS NOT INITIALIZED

2.30.3 Size

Approximate number of source statements (excluding comments): 55

Approximate size of object code in bytes (Amdahl): 2,500

2.30.4 Restrictions

Use zero in parameter 9 if one standard parallel; otherwise use a non-zero value.

2.31 SUBROUTINE PJ09Z0

Projection: Transverse Mercator

2.31.1 Definition of Parameter Array

1 Semimajor axis of spheroid
2 Eccentricity squared OR semiminor axis of spheroid
3 Scale factor at central meridian
4 Not used
5 Longitude at central meridian
6 Latitude at origin of projection
7 False easting applied to all coordinates
8 False northing applied to all coordinates
9 Not used
10 Not used
11 Not used
12 Not used
13 Not used
14 Not used
15 Not used

2.31.2 Error Messages

092 Forward: PROJECTION WAS NOT INITIALIZED

093 Forward: POINT PROJECTS INTO INFINITY

094 Inverse: PROJECTION WAS NOT INITIALIZED

095 Inverse: LATITUDE FAILED TO CONVERGE AFTER . . . ITERATIONS

2.31.3 Size

Approximate number of source statements (excluding comments): 119

Approximate size of object code in bytes (Amdahl): 5,968

2.31.4 Restrictions

The computations on the ellipsoid are valid within about 0.1 radians (about 5.7°) of longitude from the central meridian. The formulas break down very rapidly as the computations get further from that meridian.

2.32 SUBROUTINE PJ10Z0

Projection: Stereographic

2.32.1 Definition of Parameter Array

1 Radius of sphere of reference
2 Not used
3 Not used
4 Not used
5 Longitude at center of projection
6 Latitude at center of projection
7 False easting applied to all coordinates
8 False northing applied to all coordinates
9 Not used
10 Not used
11 Not used
12 Not used
13 Not used
14 Not used
15 Not used

2.32.2 Error Messages

102 Forward: PROJECTION WAS NOT INITIALIZED

103 Forward: POINT PROJECTS INTO INFINITY

104 Inverse: PROJECTION WAS NOT INITIALIZED

2.32.3 Size

Approximate number of source statements (excluding comments): 79

Approximate size of object code in bytes (Amdahl): 3,492

2.32.4 Restrictions

None

2.33 SUBROUTINE PJ11Z0

Projection: Lambert Azimuthal Equal-Area

2.33.1 Definition of Parameter Array

1 Radius of sphere of reference
2 Not used
3 Not used
4 Not used
5 Longitude of center of projection
6 Latitude of center of projection
7 False easting applied to all coordinates
8 False northing applied to all coordinates
9 Not used
10 Not used
11 Not used
12 Not used
13 Not used
14 Not used
15 Not used

2.33.2 Error Messages

112 Forward: PROJECTION WAS NOT INITIALIZED

113 Forward: POINT PROJECTS INTO A CIRCLE OF RADIUS . . . METERS

114 Inverse: PROJECTION WAS NOT INITIALIZED

115 Inverse: INPUT DATA ERROR

2.33.3 Size

Approximate number of source statements (excluding comments): 86

Approximate size of object code in bytes (Amdahl): 3,672

2.33.4 Restrictions

None

2.34 SUBROUTINE PJ12Z0

Projection: Azimuthal Equidistant

2.34.1 Definition of Parameter Array

1 Radius of sphere of reference
2 Not used
3 Not used
4 Not used
5 Longitude of center of projection
6 Latitude of center of projection
7 False easting applied to all coordinates
8 False northing applied to all coordinates
9 Not used
10 Not used
11 Not used
12 Not used
13 Not used
14 Not used
15 Not used

2.34.2 Error Messages

122 Forward: PROJECTION WAS NOT INITIALIZED

123 Forward: POINT PROJECTS INTO A CIRCLE OF RADIUS . . . METERS

124 Inverse: PROJECTION WAS NOT INITIALIZED

125 Inverse: INPUT DATA ERROR

2.34.3 Size

Approximate number of source statements (excluding comments): 88

Approximate size of object code in bytes (Amdahl): 3,732

2.34.4 Restrictions

None

2.35 SUBROUTINE PJ13Z0

Projection: Gnomonic

2.35.1 Definition of Parameter Array

1 Radius of sphere of reference
2 Not used
3 Not used
4 Not used
5 Longitude of center of projection
6 Latitude of center of projection
7 False easting applied to all coordinates
8 False northing applied to all coordinates
9 Not used
10 Not used
11 Not used
12 Not used
13 Not used
14 Not used
15 Not used

2.35.2 Error Messages

132 Forward: PROJECTION WAS NOT INITIALIZED

133 Forward: POINT PROJECTS INTO INFINITY

134 Inverse: PROJECTION WAS NOT INITIALIZED

2.35.3 Size

Approximate number of source statements (excluding comments): 79

Approximate size of object code in bytes (Amdahl): 3,408

2.35.4 Restrictions

None

2.36 SUBROUTINE PJ14Z0

Projection: Orthographic

2.36.1 Definition of Parameter Array

1 Radius of sphere of reference
2 Not used
3 Not used
4 Not used
5 Longitude of center of projection
6 Latitude of center of projection
7 False easting applied to all coordinates
8 False northing applied to all coordinates
9 Not used
10 Not used
11 Not used
12 Not used
13 Not used
14 Not used
15 Not used

2.36.2 Error Messages

142 Forward: PROJECTION WAS NOT INITIALIZED

143 Forward: POINT CANNOT BE PROJECTED

144 Inverse: PROJECTION WAS NOT INITIALIZED

145 Inverse: INPUT DATA ERROR

2.36.3 Size

Approximate number of source statements (excluding comments): 84

Approximate size of object code in bytes (Amdahl): 3,568

2.36.4 Restrictions

None

2.37 SUBROUTINE PJ15Z0

Projection: General Vertical Near-Side Perspective

2.37.1 Definition of Parameter Array

1 Radius of sphere of reference
2 Not used
3 Height of perspective point above the surface of the sphere
4 Not used
5 Longitude of center of projection
6 Latitude of origin of projection
7 False easting applied to all coordinates
8 False northing applied to all coordinates
9 Not used
10 Not used
11 Not used
12 Not used
13 Not used
14 Not used
15 Not used

2.37.2 Error Messages

152 Forward: PROJECTION WAS NOT INITIALIZED

153 Forward: POINT CANNOT BE PROJECTED

154 Inverse: PROJECTION WAS NOT INITIALIZED

155 Inverse: INPUT DATA ERROR

2.37.3 Size

Approximate number of source statements (excluding comments): 88

Approximate size of object code in bytes (Amdahl): 3,816

2.37.4 Restrictions

None

2.38 SUBROUTINE PJ16Z0

Projection: Sinusoidal

2.38.1 Definition of Parameter Array

1 Radius of sphere of reference
2 Not used
3 Not used
4 Not used
5 Longitude of central meridian
6 Not used
7 False easting applied to all coordinates
8 False northing applied to all coordinates
9 Not used
10 Not used
11 Not used
12 Not used
13 Not used
14 Not used
15 Not used

2.38.2 Error Messages

162 Forward: PROJECTION WAS NOT INITIALIZED

163 Inverse: PROJECTION WAS NOT INITIALIZED

164 Inverse: INPUT DATA ERROR

2.38.3 Size

Approximate number of source statements (excluding comments): 56

Approximate size of object code in bytes (Amdahl): 2,208

2.38.4 Restrictions

None

2.39 SUBROUTINE PJ17Z0

Projection: Equirectangular

2.39.1 Definition of Parameter Array

1 Radius of sphere of reference
2 Not used
3 Not used
4 Not used
5 Longitude at central meridian
6 Latitude of true scale
7 False easting applied to all coordinates
8 False northing applied to all coordinates
9 Not used
10 Not used
11 Not used
12 Not used
13 Not used
14 Not used
15 Not used

2.39.2 Error Messages

171 Initialization: PROJECTION WAS NOT INITIALIZED

172 Forward: PROJECTION WAS NOT INITIALIZED

173 Inverse: PROJECTION WAS NOT INITIALIZED

174 Inverse: INPUT DATA ERROR

2.39.3 Size

Approximate number of source statements (excluding comments): 49

Approximate size of object code in bytes (Amdahl): 2,048

2.39.4 Restrictions

None

2.40 SUBROUTINE PJ18Z0

Projection: Miller Cylindrical

2.40.1 Definition of Parameter Array

1 Radius of sphere of reference
2 Not used
3 Not used
4 Not used
5 Longitude at central meridian
6 Not used
7 False easting applied to all coordinates
8 False northing applied to all coordinates
9 Not used
10 Not used
11 Not used
12 Not used
13 Not used
14 Not used
15 Not used

2.40.2 Error Messages

182 Forward: PROJECTION WAS NOT INITIALIZED

183 Inverse: PROJECTION WAS NOT INITIALIZED

2.40.3 Size

Approximate number of source statements (excluding comments): 45

Approximate size of object code in bytes (Amdahl): 2,032

2.40.4 Restrictions

None

2.41 SUBROUTINE PJ19Z0

Projection: Van der Grinten

2.41.1 Definition of Parameter Array

1 Radius of sphere of reference
2 Not used
3 Not used
4 Not used
5 Longitude at central meridian
6 Not used
7 False easting applied to all coordinates
8 False northing applied to all coordinates
9 Not used
10 Not used
11 Not used
12 Not used
13 Not used
14 Not used
15 Not used

2.41.2 Error Messages

192 Forward: PROJECTION WAS NOT INITIALIZED

193 Inverse: PROJECTION WAS NOT INITIALIZED

2.41.3 Size

Approximate number of source statements (excluding comments): 91

Approximate size of object code in bytes (Amdahl): 4,124

2.41.4 Restrictions

None

2.42 SUBROUTINE PJ20Z0

Projection: Oblique Mercator (Hotine)

2.42.1 Definition of Parameter Array

	Format A		Format B
1	Semimajor axis of spheroid	OR	same
2	Eccentricity squared semiminor axis	OR	same
3	Scale factor at center	OR	same
4	Not used	OR	azimuth east of north for central line
5	Not used	OR	long. of point of origin
6	Latitude of projection origin	OR	same
7	False easting	OR	same
8	False northing	OR	same
9	Longitude of first point defining central line	OR	not used
10	Latitude of first point	OR	not used
11	Longitude of second point	OR	not used
12	Latitude of second point	OR	not used
13	Zero	OR	non-zero
14	Not used		
15	Not used		

2.42.2 Error Messages

204 Forward: PROJECTION WAS NOT INITIALIZED

205 Forward: POINT PROJECTS INTO INFINITY

206 Inverse: PROJECTION WAS NOT INITIALIZED

2.42.3 Size

Approximate number of source statements (excluding comments): 89

Approximate size of object code in bytes (Amdahl): 3,992

2.42.4 Restrictions

None

2.43 SUBROUTINE PJ21Z0

Projection: Robinson

2.43.1 Definition of Parameter Array

1 Radius of sphere of reference
2 Not used
3 Not used
4 Not used
5 Longitude at central meridian
6 Not used
7 False easting applied to all coordinates
8 False northing applied to all coordinates
9 Not used
10 Not used
11 Not used
12 Not used
13 Not used
14 Not used
15 Not used

2.43.2 Error Messages

212 Forward: PROJECTION WAS NOT INITIALIZED

213 Inverse: PROJECTION WAS NOT INITIALIZED

2.43.3 Size

Approximate number of source statements (excluding comments): 80

Approximate size of object code in bytes (Amdahl): 3,892

2.43.4 Restrictions

None

2.44 SUBROUTINE PJ22Z0

Projection: Space Oblique Mercator

2.44.1 Definition of Parameter Array

1 Semimajor axis of spheroid
2 Eccentricity squared OR semiminor axis of spheroid
3 Landsat number (1 to 5)
4 Path number (1-251 for Landsat 1-3 OR 1-233 for Landsat 4-5)
5 Not used
6 Not used
7 False easting applied to all coordinates
8 False northing applied to all coordinates
9 Not used
10 Not used
11 Not used
12 Not used
13 Not used
14 Not used
15 Not used

2.44.2 Error Messages

222 Forward: PROJECTION WAS NOT INITIALIZED

223 Inverse: 50 ITERATIONS WITHOUT CONVERGENCE

224 Inverse: PROJECTION WAS NOT INITIALIZED

2.44.3 Size

Approximate number of source statements (excluding comments): 133

Approximate size of object code in bytes (Amdahl): 6,836

2.44.4 Restrictions

Landsat 1 through 5

2.45 SUBROUTINE PJ23Z0

Projection: Modified Stereographic Conformal (for Alaska)

2.45.1 Definition of Parameter Array

1 Semimajor axis of Clarke 1866 spheroid at map scale
2 Eccentricity squared of Clarke 1866 spheroid
3 Not used
4 Not used
5 Not used
6 Not used
7 False easting applied to all coordinates
8 False northing applied to all coordinates
9 Not used
10 Not used
11 Not used
12 Not used
13 Not used
14 Not used
15 Not used

2.45.2 Error Messages

232 Forward: PROJECTION WAS NOT INITIALIZED

234 Inverse: PROJECTION WAS NOT INITIALIZED

235 Inverse: TOO MANY ITERATIONS IN ITERATING INVERSE

236 Inverse: TOO MANY ITERATIONS IN CALCULATING PHI FROM CHI

2.45.3 Size

Approximate number of source statements (excluding comments): 149

Approximate size of object code in bytes (Amdahl): 6,128

2.45.4 Restrictions

Alaska only

2.46 FUNCTION QSFNZ0

DOUBLE PRECISION FUNCTION QSFNZ0(ECC,SINPHI,COSPHI)

2.46.1 Summary

This function computes the constant "q" (variable QS) used in the forward computation for Albers Conical Equal-Area projection. All variables are REAL*8.

ECC = Eccentricity of spheroid
SINPHI = Sine of the latitude sin(PHI)
COSPHI = Cosine of the latitude cos(PHI)

2.46.2 Algorithm

$$QS = (1-ECC^2) * \left[\frac{SINPHI}{(1-ECC^2 * SINPHI^2)} - \frac{1}{2 * ECC} * \ln \frac{(1-ECC * SINPHI)}{(1+ECC * SINPHI)}\right]$$

Note: ln is the natural log (base e).

2.46.3 Error Messages

None

2.46.4 Size

Approximate number of source statements (excluding comments): 11

Approximate size of object code in bytes (Amdahl): 1,004

2.47 SUBROUTINE RADDZ0

SUBROUTINE RADDZ0(RAD,SGNA,DEGS,MINS,SECS)

2.47.1 Summary

This subroutine converts a REAL*8 angle RAD in radians to sign SGNA, degrees DEGS, minutes MINS, and seconds SECS.

- SGNA is the sign as either blank (for positive) or a minus sign (for negative), stored as a CHARACTER*1 value.
- DEGS is the degrees, stored as an INTEGER*4 value.
- MINS is the minutes, stored as an INTEGER*4 value.
- SECS is the seconds, stored as a REAL*4 value.

2.47.2 Algorithm

(1) If RAD is positive, set SGNA to a blank (' '); if negative, set SGNA to a minus sign ('-').

(2) Convert radians to seconds: CON = RAD * 206264.80625

(3) Divide by 3600 and truncate to get degrees; put in DEGS

(4) Subtract 3600 * DEGS from CON; divide by 60 and truncate to get minutes; put in MINS.

(5) Remainder from step (4) is seconds; put in SECS.

(6) If SECS is larger than 59.9995, MINS is increased by 1 and SECS is set to zero.

(7) If MINS is larger than 59, DEGS is increased by 1 and MINS is set to zero.

2.47.3 Error Messages

None

2.47.4 Size

Approximate number of source statements (excluding comments): 29

Approximate size of object code in bytes (Amdahl): 1,404

2.48 SUBROUTINE SERAZ0

SUBROUTINE SERAZ0(FB,FA2,FA4,FC1,FC3,LAM)

2.48.1 Summary

This subroutine computes the integral function of the transformed longitude LAM for Fourier constants FB, FA2, FA4, FC1, and FC3 in initialization of the Space Oblique Mercator projection. All variables are REAL*8.

```
DG1 = 0.0174329252D0
LAM = Value of the transformed longitude in deci-degrees
SA  = sine of the inclination
CA  = cosine of the inclination
P22 = revolution time for satellite / rotation time for Earth
```

2.48.2 Algorithm

```
LAM = LAM * DG1
SD = SIN(LAM)
SDSQ = SD * SD
S = P22 * SA * COS(LAM) * SQRT( (1 + T * SDSQ) / ( (1 + W * SDSQ)
    * (1 + Q * SDSQ) ) )
H = SQRT( (1 + Q * SDSQ) / (1 + W * SDSQ) ) * ( ( (1 + W * SDSQ)
    / ( (1 + Q * SDSQ) ** 2) ) - P22 * CA)
SQ = SQRT(XJ * XJ + S * S)
FB = (H * XJ - S * S) / SQ
FA2 = FB * COS(2 * LAM)
FA4 = FB * COS(4 * LAM)
FC = S * (H + XJ) / SQ
FC1 = FC * COS(LAM)
FC3 = FC * COS(3 * LAM)
```

2.48.3 Error Messages

None

2.48.4 Size

Approximate number of source statements (excluding comments): 19

Approximate size of object code in bytes (Amdahl): 1,788

2.49 SUBROUTINE SPHDZ0

SUBROUTINE SPHDZ0(ISPH,PARM)

2.49.1 Summary

Subroutine SPHDZ0 performs the input of the spheroid values of semimajor axis and the eccentricity squared into the parameter array by the use of an INTEGER*4 code number ISPH. In addition, the default reference spheroid can be reset, if desired, to one of the possible spheroids. If ISPH is negative, the user-specified spheroid parameters in variables PARM(1) and PARM(2) are used to define the radius of sphere or spheroid semimajor axis A, and semiminor axis B or eccentricity squared ES. ES is computed if B is provided greater than one. If B is zero, a sphere is assumed. If A and B are zero, the Clarke 1866 spheroid is assumed. If B is specified and A is not, the Clarke 1866 spheroid semimajor axis is assumed and is associated with B. All variables are REAL*8 except ISPH.

PARM	Array of 15 projection parameters
A = AZ	Semimajor axis of spheroid in meters
B	Semiminor axis of spheroid or eccentricity squared
AZZ	Radius of sphere
EZ	Eccentricity of spheroid (zero if sphere)
ES = ESZ	Eccentricity squared (zero if sphere)

2.49.2 Algorithm

PARM(1) is the semimajor axis A

PARM(2) is the eccentricity squared ES, computed from the semimajor axis and semiminor axis B, if it is greater than one, as follows:

$$ES = 1 - (B / A)^2$$

The ellipsoid constants are computed as follows:

E0Z = E0FNZ0(ES)

E1Z = E1FNZ0(ES)

E2Z = E2FNZ0(ES)

E3Z = E3FNZ0(ES)

E4Z = E4FNZ0(EZ)

ISPH is the spheroid code as follows:

Code	Spheroid	Code	Spheroid
0	Clarke 1866	10	Modified Everest
1	Clarke 1880	11	Modified Airy
2	Bessel	12	WGS 84
3	New International 1967	13	Southeast Asia
4	International 1909	14	Australian National
5	WGS 72	15	Krassovsky
6	Everest	16	Hough
7	WGS 66	17	Mercury 1960
8	GRS 1980	18	Modified Mercury 1968
9	Airy	19	Normal Sphere

2.49.3 Error Messages

The following is printed if a code ISPH is greater than 19:

SPHEROID CODE OF . . . RESET TO 0

2.49.4 Size

Approximate number of source statements (excluding comments): 73

Approximate size of object code in bytes (Amdahl): 2,812

2.50 FUNCTION TSFNZ0

DOUBLE PRECISION FUNCTION TSFNZ0(ECC,PHI,SINPHI)

2.50.1 Summary

This function computes the constant "t" for use in the forward computations in the Lambert Conformal Conic and the Polar Stereographic projections. All variables are REAL*8.

ECC = Eccentricity of the spheroid

PHI = latitude phi

SINPHI = sine of the latitude sin(PHI)

PI = the constant PI

2.50.2 Algorithm

$$t = \mathrm{TAN}(PI/4 - PHI/2) * \left[\frac{(1 + ECC * SINPHI)}{(1 - ECC * SINPHI)}\right]^{ECC/2}$$

2.50.3 Error Messages

None

2.50.4 Size

Approximate number of source statements (excluding comments): 10

Approximate size of object code in bytes (Amdahl): 924

2.51 SUBROUTINE UNTFZ0

SUBROUTINE UNTFZ0(INUNIT,IOUNIT,FACTOR,IFLG)

2.51.1 Summary

This subroutine determines the FACTOR as REAL*8 to multiply between two lineal unit types where:

INUNIT is the code for the input units.

IOUNIT is the code for the output or target units.

FACTOR is the multiplier determined by the subroutine.

IFLG is the error flag.

2.51.2 Algorithm

The following codes are used:

0 = Radians
1 = U.S. feet
2 = Meters
3 = Seconds of arc
4 = Degrees of arc
5 = International feet

EXAMPLES: INUNIT = 1; IOUNIT = 2; FACTOR = .3048006096012192
INUNIT = 4; IOUNIT = 3; FACTOR = 3600

2.51.3 Error Messages

ILLEGAL SOURCE OR TARGET UNIT CODE = . . . / . . . and IFLG = 11

INCONSISTENT UNIT CODES = . . . / . . . and IFLG = 12, when conversion was specified between distance and angular units or vice versa.

2.51.4 Size

Approximate number of source statements (excluding comments): 29

Approximate size of object code in bytes (Amdahl): 1,788

3. OPERATING ENVIRONMENT

3.1 HARDWARE

The master version of the GCTP is operational on the Amdahl 5890/300E computer. It has been compiled using the IBM VS FORTRAN compiler and is generally used as object modules linked to application programs. The package has been compiled without change on other systems such as the Concurrent 3280, Gould Concept 9780, and Digital VAX 11/750 computers.

3.2 SUPPORT SOFTWARE

3.2.1 Operating System

The GCTP has been tested on the Amdahl 5890/300E computer operating under IBM MVS/XA JES2, the Concurrent 3280 operating under OS/32, the Gould Concept 9780 operating under MPX-32, and the Digital VAX 11/750 operating under VMS. No machine or operating system-dependent features are used.

3.2.2 Compiler/Interpreter/Assembler

The routines were written in FORTRAN under the 1977 ANSI standard. Installation with compilers not completely compatible with ANSI FORTRAN 1977 may require minor modifications.

3.2.3 Other Software

No software, other than the calling program, is required except for the normal FORTRAN compiler run-time libraries.

3.3 DATA BASE

The GCTP requires two direct access files containing the State Plane zone parameters for NAD 1927 and NAD 1983 coordinate transformations. These files may be built with program SPLOAD, which converts two ASCII files, containing 134 zones of 4 records each, to direct access files for GCTP. Program SPLOAD, the NAD 1927 file, and the NAD 1983 file are supplied with software for GCTP.

4. MAINTENANCE PROCEDURES

4.1 PROGRAMMING CONVENTIONS

Standard ANSI FORTRAN programming conventions have been followed. There is extensive use of double precision (REAL*8) variables and functions.

No utility programs or other subroutines are called other than the following FORTRAN run-time routines:

DABS, DBLE, DCOS, DSIN, DTAN, DSQRT, DEXP, DLOG, DACOS, DASIN, DATAN, DATAN2, DMAX1, DMIN1, DMOD, DSIGN, IABS, IDINT, and SNGL.

4.2 VERIFICATION PROCEDURES

See the appendixes to USGS Professional Paper 1395 for sample data.

4.3 ERROR CORRECTION PROCEDURES

Any suspected software errors should be documented on a Discrepancy Report form (p. 70-71) and forwarded to the office cited in section 4.7.

4.4 SPECIAL MAINTENANCE PROCEDURES

No special procedures are required.

4.5 LISTING AND CHARTS

No flowcharts are available. Source code may be obtained from the maintenance office listed in section 4.6.

4.6 POINT OF CONTACT

U.S. Geological Survey
National Mapping Division
Office of Production Operations
Branch of Operations Policy
National Center Mail Stop 510
Reston, Virginia 22092

4.7 DISCREPANCY REPORTS

Reports of corrections and suggested modifications should be sent on a Discrepancy Report form (p. 70-71) to:

U.S. Geological Survey
National Mapping Division
Office of Production Operations
Configuration Management Office
National Center Mail Stop 510
Reston, Virginia 22092

DISCREPANCY REPORT / STATEMENT OF NEED			DR #: SON #:
Name:	Office:	Phone:	Date:
Configuration Item and Number:		Priority:	
Type ___ hardware ___ database ___ software ___ facility ___ document ___ other:		CIs impacted:	
Date Occurred:		Activity in Progress:	
Location: __MAC __WMC __RMMC __EDC __MCMC __HQ __Other:			
Supervisor's Signature:			
Description:			
Date fix required by:		Technical Investigation Assigned to:	
Date logged: Suspense Date: Action: ___ Reject ___ Withdraw ___ Close ___ RFC TI signature:			
Verified by:	Quality Assurance:	Date:	
Keywords:			

This sheet represents page 71, which is blank.

APPENDIXES A-C

Appendix A. State Plane Coordinate Systems

Jurisdiction zone name or number	Alpha code		Proj. type	NAD27 zone code	NAD83 zone code
Alabama	AL				
East			TM	0101	0101
West			TM	0102	0102
Alaska	AK				
01			OM	5001	5001
02			TM	5002	5002
03			TM	5003	5003
04			TM	5004	5004
05			TM	5005	5005
06			TM	5006	5006
07			TM	5007	5007
08			TM	5008	5008
09			TM	5009	5009
10			LB	5010	5010
Arizona	AZ				
East			TM	0201	0201
Central			TM	0202	0202
West			TM	0203	0203
Arkansas	AR				
North			LB	0301	0301
South			LB	0302	0302
California	CA				
01			LB	0401	0401
02			LB	0402	0402
03			LB	0403	0403
04			LB	0404	0404
05			LB	0405	0405
06			LB	0406	0406
07			LB	0407	----
Colorado	CO				
North			LB	0501	0501
Central			LB	0502	0502
South			LB	0503	0503
Connecticut	CT		LB	0600	0600
Delaware	DE		TM	0700	0700
District of Columbia	DC	use	LB	1900	1900
Florida	FL				
East			TM	0901	0901
West			TM	0902	0902
North			LB	0903	0903

Appendix A. State Plane Coordinate Systems--Continued

Jurisdiction zone name or number	Alpha code	Proj. type	NAD27 zone code	NAD83 zone code
Georgia	GA			
East		TM	1001	1001
West		TM	1002	1002
Hawaii	HI			
01		TM	5101	5101
02		TM	5102	5102
03		TM	5103	5103
04		TM	5104	5104
05		TM	5105	5105
Idaho	ID			
East		TM	1101	1101
Central		TM	1102	1102
West		TM	1103	1103
Illinois	IL			
East		TM	1201	1201
West		TM	1202	1202
Indiana	IN			
East		TM	1301	1301
West		TM	1302	1302
Iowa	IA			
North		LB	1401	1401
South		LB	1402	1402
Kansas	KS			
North		LB	1501	1501
South		LB	1502	1502
Kentucky	KY			
North		LB	1601	1601
South		LB	1602	1602
Louisiana	LA			
North		LB	1701	1701
South		LB	1702	1702
Offshore		LB	1703	1703
Maine	ME			
East		TM	1801	1801
West		TM	1802	1802
Maryland	MD	LB	1900	1900
Massachusetts	MA			
Mainland		LB	2001	2001
Island		LB	2002	2002

Appendix A. State Plane Coordinate Systems--Continued

Jurisdiction zone name or number	Alpha code	Proj. type	NAD27 zone code	NAD83 zone code
Michigan	MI			
East	(Obsolete)	TM	2101	----
Central	(Obsolete)	TM	2102	----
West	(Obsolete)	TM	2103	----
North		LB	2111	2111
Central		LB	2112	2112
South		LB	2113	2113
Minnesota	MN			
North		LB	2201	2201
Central		LB	2202	2202
South		LB	2203	2203
Mississippi	MS			
East		TM	2301	2301
West		TM	2302	2302
Missouri	MO			
East		TM	2401	2401
Central		TM	2402	2402
West		TM	2403	2403
Montana	MT	LB	----	2500
North		LB	2501	----
Central		LB	2502	----
South		LB	2503	----
Nebraska	NE	LB	----	2600
North		LB	2601	----
South		LB	2602	----
Nevada	NV			
East		TM	2701	2701
Central		TM	2702	2702
West		TM	2703	2703
New Hampshire	NH	TM	2800	2800
New Jersey	NJ	TM	2900	2900
New Mexico	NM			
East		TM	3001	3001
Central		TM	3002	3002
West		TM	3003	3003

Appendix A. State Plane Coordinate Systems--Continued

Jurisdiction zone name or number	Alpha code	Proj. type	NAD27 zone code	NAD83 zone code
New York	NY			
East		TM	3101	3101
Central		TM	3102	3102
West		TM	3103	3103
Long Island		LB	3104	3104
North Carolina	NC	LB	3200	3200
North Dakota	ND			
North		LB	3301	3301
South		LB	3302	3302
Ohio	OH			
North		LB	3401	3401
South		LB	3402	3402
Oklahoma	OK			
North		LB	3501	3501
South		LB	3502	3502
Oregon	OR			
North		LB	3601	3601
South		LB	3602	3602
Pennsylvania	PA			
North		LB	3701	3701
South		LB	3702	3702
Rhode Island	RI	TM	3800	3800
South Carolina	SC	LB	----	3900
North		LB	3901	----
South		LB	3902	----
South Dakota	SD			
North		LB	4001	4001
South		LB	4002	4002
Tennessee	TN	LB	4100	4100
Texas	TX			
North		LB	4201	4201
North Central		LB	4202	4202
Central		LB	4203	4203
South Central		LB	4204	4204
South		LB	4205	4205
Utah	UT			
North		LB	4301	4301
Central		LB	4302	4302
South		LB	4303	4303
Vermont	VT	TM	4400	4400

Appendix A. State Plane Coordinate Systems--Continued

Jurisdiction zone name or number	Alpha code	Proj. type	NAD27 zone code	NAD83 zone code
Virginia	VA			
North		LB	4501	4501
South		LB	4502	4502
Washington	WA			
North		LB	4601	4601
South		LB	4602	4602
West Virginia	WV			
North		LB	4701	4701
South		LB	4702	4702
Wisconsin	WI			
North		LB	4801	4801
Central		LB	4802	4802
South		LB	4803	4803
Wyoming	WY			
East (01)		TM	4901	4901
East Central (02)		TM	4902	4902
West Central (03)		TM	4903	4903
West (04)		TM	4904	4904
Puerto Rico	PR	LB	5201	5200
Virgin Islands	VI	LB	5201	5200
St. Croix		LB	5202	5200
American Samoa	AS	LB	5300	----
Guam	GU	PC	5400	----

TM = Transverse Mercator
OM = Oblique Mercator
PC = Polyconic
LB = Lambert

Appendix B. Universal Transverse Mercator zone locations and central meridians

Zone	C.M.	Range	Zone	C.M.	Range
01	177W	180W-174W	31	003E	000E-006E
02	171W	174W-168W	32	009E	006E-012E
03	165W	168W-162W	33	015E	012E-018E
04	159W	162W-156W	34	021E	018E-024E
05	153W	156W-150W	35	027E	024E-030E
06	147W	150W-144W	36	033E	030E-036E
07	141W	144W-138W	37	039E	036E-042E
08	135W	138W-132W	38	045E	042E-048E
09	129W	132W-126W	39	051E	048E-054E
10	123W	126W-120W	40	057E	054E-060E
11	117W	120W-114W	41	063E	060E-066E
12	111W	114W-108W	42	069E	066E-072E
13	105W	108W-102W	43	075E	072E-078E
14	099W	102W-096W	44	081E	078E-084E
15	093W	096W-090W	45	087E	084E-090E
16	087W	090W-084W	46	093E	090E-096E
17	081W	084W-078W	47	099E	096E-102E
18	075W	078W-072W	48	105E	102E-108E
19	069W	072W-066W	49	111E	108E-114E
20	063W	066W-060W	50	117E	114E-120E
21	057W	060W-054W	51	123E	120E-126E
22	051W	054W-048W	52	129E	126E-132E
23	045W	048W-042W	53	135E	132E-138E
24	039W	042W-036W	54	141E	138E-144E
25	033W	036W-030W	55	147E	144E-150E
26	027W	030W-024W	56	153E	150E-156E
27	021W	024W-018W	57	159E	156E-162E
28	015W	018W-012W	58	165E	162E-168E
29	009W	012W-006W	59	171E	168E-174E
30	003W	006W-000E	60	177E	174E-180W

UTM zone numbers in the Southern Hemisphere are indicated by a negative sign before the zone number.

Example: Zone -17 has a central meridian of 81° W and a false northing (Y) of 10,000,000 meters at the Equator.

Appendix C. Parameters required for definition of map projections

Parm No.	(00) Geographic	(01)* Universal Transverse Mercator	(02)* State Plane Coordinates	(03) Albers Conical Equal-Area	(04) Lambert Conformal Conic
1	---	Longitude of any point in the UTM zone	---	Semimajor axis If blank / 0, Clarke 1866. in meters is assumed.	
2	---	Latitude of any point in the UTM zone	---	Eccentricity squared of ellipsoid. 0 for sphere If >1, semiminor axis	
3	---	---	---	Latitude of 1st standard parallel	
4	---	---	---	Latitude of 2nd standard parallel	
5	---	---	---	Longitude of the central meridian	
6	---	---	---	Latitude of the origin of projection	
7	---	---	---	False easting in units of semimajor axis	
8	---	---	---	False northing in units of semimajor axis.	

Note: Parameters 9-15 are not used for projections on this page. All angles (latitudes, longitudes, or azimuth) are required in degrees, minutes, and seconds of arc in the packed real number format ±DDDMMMSSS.SSS where ± is the sign, DDD is the degrees, MMM is the minutes, and SSS.SSS is the seconds.

* If a UTM or State Plane zone is specified, the projection parameters will be supplied by the program and those given by the user will be ignored.

Appendix C. Parameters required for definition of map projections--Continued

Parm No.	(05) Mercator	(06) Polar Stereographic	(07) Polyconic	(08) Equidistant Conic Type A	(08) Equidistant Conic Type B
1	Semimajor axis of ellipsoid. If blank or 0, Clarke 1866 in meters used.				
2	Eccentricity squared of ellipsoid (e^2). If blank or 0, assume sphere. If >1, interpreted as semiminor axis of ellipsoid.				
3	---	---	---	Latitude of standard parallel	Latitude of 1st standard parallel
4	---	---	---	---	Latitude of 2nd standard parallel
5	Longitude of central meridian	Longitude straight down from North Pole, up from South Pole	Longitude of central meridian		
6	Latitude of true scale	Latitude of true scale	Latitude of origin of projection		
7	False easting in the same units as the semimajor axis				
8	False northing in the same units as the semimajor axis				
9	---	---	---	zero	any non-0 number

Note: Parameters 10-15 are not used for projections on this page.

Appendix C. Parameters required for definition of map projections--Continued

Parm No.	(09) Transverse Mercator	(10) Stereo-graphic	(11) Lamb. Azimuth. Equal-Area	(12) Azimuthal Equidistant	(13) Gnomonic
1	Same as (03) to (08)	Radius of sphere of reference (Default of 6370997 meters)			
2	Same as (03) to (08)	---	---	---	---
3	Scale factor at central meridian	---	---	---	---
4	---	---	---	---	---
5	Longitude of central meridian	Longitude of center of projection			
6	Latitude of origin	Latitude of center of projection			
7	False easting in the same units as the semimajor axis or radius of the sphere				
8	False northing in the same units as the semimajor axis or radius of the sphere				

Note: Parameters 9-15 are not used for projections on this page.

Appendix C. Parameters required for definition of map projections--Continued

Parm No.	(14) Orthographic	(15) General Vert Near-Side Perspective	(16) Sinusoidal	(17) Equi-rectangular	(18) Miller Cylindrical
1	Radius of sphere of reference (default of 6370997 meters)				
2	---	---	---	---	---
3	---	Height of perspective point above surface	---	---	---
4	---	---	---	---	---
5	Longitude of center of projection		Longitude of central meridian		
6	Latitude of center of projection		---	Latitude of true scale	---
7	False easting in the same units as the radius of sphere				
8	False northing in the same units as the radius of sphere				

Note: Parameters 9-15 are not used for projections on this page.

Appendix C. Parameters required for definition of map projections--Continued

Parm No.	(19) Van der Grinten	(20) Oblique Mercator Format A	Format B	No.	Format A	Format B
1	Radius of reference sphere	Semimajor axis of ellipsoid (default Clarke 1866)		9	Longitude 1st point on center line	---
2	---	Eccentricity squared (0 for sphere). >1 for semiminor axis		10	Latitude 1st point on center line	---
3	---	Scale factor at center of the projection		11	Longitude 2nd point on center line	---
4	---	---	Az. angle E of N of center line.	12	Latitude 2nd point on center line	---
5	Longitude of central meridian	---	Longitude point on line where azimuth is measured	13	zero	Any non-zero number
6	---	Latitude of the origin of the projection		14	---	---
7	False east in units of radius	False easting in the same units as the semimajor axis		15	---	---
8	False north in units of radius	False northing in the same units as the semimajor axis			---	---

Note: Parameters 9-15 are not used for Van der Grinten projection.

Appendix C. Parameters required for definition of map projections--Continued

Parm No.	(21) Robinson	(22) Space Oblique Mercator	(23) Modified Stereographic for Alaska
1	Radius of sphere of reference	Semimajor axis of ellipsoid. If blank or 0, Clarke 1866 in meters used.	Semimajor axis of Clarke 1866 ellipsoid must be used
2	---	Eccentricity squared (0 for sphere). If >1, semiminor axis	Eccentricity squared of Clarke 1866 ellipsoid must be used
3	---	Landsat number	---
4	---	Path number	---
5	Longitude at central meridian	---	---
6	---	---	---
7	False easting in the same units as the semimajor axis		
8	False northing in the same units as the semimajor axis		

Note: Parameters 9-15 are not used for projections on this page.