REUSE MANUAL

DATUM

10xxxxxx.1

Implementation

TABLE OF CONTENTS

SECTION 1. INTRODUCTION

1.1 PURPOSE OF THE REUSE MANUAL

This document describes the characteristics of the DATUM reusable software component and provides instructions on its installation and operation. The manual is a self-contained reference for the software engineer intending to incorporate the component in another software system. This manual was written with the assumption that the user has a basic working knowledge of C and is familiar with fundamental C concepts and terminology.

1.2 PURPOSE OF THE REUSABLE SOFTWARE COMPONENT

This component provides datum shifts for a large collection of local datums, WGS 72 and WGS 84. A particular datum can be accessed by using its standard 5-letter code to find its index in the datum table. The index can then be used to retrieve the datum name, type, ellipsoid code, and datum shift parameters, and to perform data shifts to or from that datum.

By sequentially retrieving all of the datum codes and/or names, a menu of the available datums can be constructed. The index values resulting from selections from this menu can then be used to access the parameters of the selected datum, or to perform datum shifts involving that datum.

This component supports both 3-parameter local datums, for which only X, Y, and Z translations relative to WGS 84 have been defined, and 7-parameter local datums, for which X, Y, and Z rotations, and a scale factor, are also defined. It also includes entries for WGS 84 (with an index of 0), and WGS 72 (with an index of 1), but no shift parameter values are defined for these.

This component provides datum shift functions for both geocentric and geodetic coordinates. WGS 84 is used as an intermediate state when shifting from one local datum to another. When geodetic coordinates are given, Molodensky’s method is normally used. Specific algorithms are used for shifts between WGS 72 and WGS 84.

This component depends on two data files, named “3_param.dat” and “7_param.dat”, which contain the datum parameter values. Copies of these files must be located in the directory specified by the value of the environment variable "DATUM_DATA", if defined, or else in the current directory, whenever a program containing this component is executed.

Additional datums can be added to these files, either manually or using the Create_Datum function. However, if a large number of datums are added, the datum table array sizes in this component will have to be increased.

This component depends on two other components: the ELLIPSOID component for access to ellipsoid parameters; and the GEOCENTRIC component for conversions between geodetic and geocentric coordinates.

1.3 GENERAL INFORMATION

1.3.1 POINT OF CONTACT

U.S. Army Topographic Engineering Center (USATEC)

Geospatial Information Division

ATTN: CETEC-GD-A (Dan Specht)

7701 Telegraph Road

Alexandria, VA 22315-3864

Dan Specht (703) 428 - 6761 Project Manager

1.3.2 CERTIFICATION LEVEL

This RSC has been certified at level 4. A level 4 component satisfies the criteria for reliability, testing, and documentation for the Army Reuse Center (ARC). The component comes with test materials and a Reuse Manual that aids in integrating the component into a software system.

1.3.3 LEGAL RESTRICTIONS

SECTION 2. INSTALLATION

The following is a list of the files which make up the DATUM component:

Source Code Files:

```
datum.c
```

Header Files :

```
datum.h
```

Data Files :

```
3_param.dat
7_param.dat
```

The compilation instructions for the DATUM component are as follows:

DOS Makefile (Uses Microsoft C):

```
cl /nologo /W3 /FR /G2 /DNDEBUG /Gs /Ox /AM /D_DOS /c datum.c
```

UNIX Makefile (Uses gcc compiler):

```
cc -g -O -ansi -Wall -c datum.c
```

The compilation order of the DATUM component relative to other components is unconstrained.

2.1 PARTIAL REUSE

The DATUM component does not allow for partial reuse.

2.2 MODIFICATIONS

The DATUM component does not permit modifications.

SECTION 3. ENVIRONMENT

This section provides details on the environment under which DATUM was developed, tested, and executed.

3.1 HARDWARE

3.1.1 DEVELOPMENT

The following is a list of hardware configurations under which DATUM was developed and tested.

- SUN SparcStation 20
- IBM compatible Pentium PC

3.1.2 TARGET

The following is a list of hardware configurations under which DATUM was executed.

- SUN SparcStation 20
- IBM compatible Pentium PC

3.2 SOFTWARE

3.2.1 OPERATING SYSTEM

The following is a list of operating systems under which DATUM was executed and tested.

- Solaris 2.5
- Windows 95

3.2.2 COMPILERS

The following is a list of compilers on which DATUM was compiled successfully.

- GCC version 2.8.1
- Microsoft Visual C++ version 6

3.3 ASSUMPTIONS AND PERFORMANCE LIMITATIONS

There are no hardware or environment constraints. There are no limitations.

This RSC is written in ANSI C.

SECTION 4. GLOBAL RSC ENVIRONMENT

4.1 TYPES

The following is a list of global type declarations in DATUM with their descriptions.

The enumerated type Datum_Type defines the different types of datums: local datums with by three datum shift parameters, local datums with by seven datum shift parameters, WGS 84, and WGS 72.

```
typedef enum Datum_Type_Code
{
   Three_Param_Datum,
   Seven_Param_Datum,
   WGS84_Datum,
   WGS72_Datum
} Datum_Type;
```

4.2 CONSTANTS

The following is a list of significant visible constants declared globally in DATUM with their descriptions.

```
DATUM_NO_ERROR                    : No errors occurred in function
DATUM_NOT_INITIALIZED_ERROR       : Datum module has not been initialized
DATUM_7PARAM_FILE_OPEN_ERROR      : 7 parameter file opening error
DATUM_7PARAM_FILE_PARSING_ERROR   : 7 parameter file structure error
DATUM_7PARAM_OVERFLOW_ERROR       : 7 parameter table overflow
DATUM_3PARAM_FILE_OPEN_ERROR      : 3 parameter file opening error
DATUM_3PARAM_FILE_PARSING_ERROR   : 3 parameter file structure error
DATUM_3PARAM_OVERFLOW_ERROR       : 3 parameter table overflow
DATUM_INVALID_INDEX_ERROR         : Index out of valid range (less than one or
                                       more than Get_Datum_Count)
DATUM_INVALID_SRC_INDEX_ERROR:    Source datum index invalid
DATUM_INVALID_DEST_INDEX_ERROR    : Destination datum index invalid
DATUM_INVALID_CODE_ERROR          : Datum code not found in table
DATUM_LAT_ERROR                   : Latitude out of valid range (-90 to 90)
DATUM_LON_ERROR                   : Longitude out of valid range (-180 to 360)
DATUM_SIGMA_ERROR                 : Standard error values must be positive (or
                                     -1 if unknown)
DATUM_DOMAIN_ERROR                : Domain of validity not well defined
DATUM_ELLIPSE_ERROR               : Error in ellipsoid module
```

4.3 VARIABLES

Not applicable. All variables declared in this component are private.

4.4 INCLUDE FILES

```
ctype.h                          : Stamdard C character handling library
math.h                           : Standard C math library
stdio.h                          : Standarc C input/output library
stdlib.h                         : Standard C general utility library
string.h                         : Standard C string handling library
ellipse.h                        : Used to retrieve ellipsoid axis
geocent.h                        : Used to convert between geocentric and
                                     geodetic coordinates
datum.h                          : Error codes and prototype error checking
```

4.5 DEPENDENCIES

The following is a list of the software external to the RSC and its descriptions.

ANSI C standard character handling, math, input/output, general utility, and string handling, libraries are used as described above,

ELLIPSOID, which provides access to a table of standard ellipsoid parameters, and

GEOCENTRIC, which performs coordinate conversions between geodetic (latitude, longitude, height) coordinates and geocentric (X, Y, Z) coordinates.

SECTION 5. FUNCTIONS

5.1 INITIALIZE_DATUMS

5.1.1 DESCRIPTION

This function initializes the module by reading in two files named “3_param.dat” and “7_param.dat” which must be in the current directory. It uses the data from these files to create a datum table.

5.1.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Initialize_Datums();
```

5.1.3 DECLARATIONS

5.1.3.1 TYPES

Not applicable.

5.1.3.2 CONSTANTS

Not applicable.

5.1.3.3 VARIABLES

Not applicable.

5.1.4 DEPENDENCIES

Assign_Datum_Row – used to copy a table entry, and

Initialize_Ellipsoids in the ELLIPSOID component – used to initialize that module.

5.1.5 ERROR HANDLING

This function returns the following status codes:

```
DATUM_NO_ERROR                   : No errors occurred in function
DATUM_ELLIPSE_ERROR              : Error in ellipsoid module
DATUM_7PARAM_FILE_OPEN_ERROR     : 7 parameter file opening error
DATUM_7PARAM_FILE_PARSING_ERROR  : 7 parameter file structure error
DATUM_7PARAM_OVERFLOW_ERROR      : 7 parameter table overflow
DATUM_3PARAM_FILE_OPEN_ERROR     : 3 parameter file opening error
DATUM_3PARAM_FILE_PARSING_ERROR  : 3 parameter file structure error
DATUM_3PARAM_OVERFLOW_ERROR      : 3 parameter table overflow
```

5.2 CREATE_DATUM

5.2.1 DESCRIPTION

This function creates a new (3-parameter) local datum with the specified code, name, ellipsoid code, and parameters, adds it to the datum table, and updates the 3-parameter datum file. The specified datum code must not already be in use. The specified ellipsoid code must be defined.

5.2.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Create_Datum ( const char *code,
                    const char *name,
                    const char *ellipsoid_code,
                    double delta_x,
                    double delta_y,
                    double delta_z,
                    double sigma_x,
                    double sigma_y,
                    double sigma_z,
                    double south_latitude,
                    double north_latitude,
                    double west_longitude,
                    double east_longitude)
```

code	The five letter code for the new datum (input),
name	The name for the new datum (input),
ellipsoid_code	The 2-letter ellipsoid code for the new datum (input),
delta_X	The X translation to WGS84 in meters for the new datum (input),
delta_Y	The Y translation to WGS84 in meters for the new datum (input),
delta_Z	The Z translation to WGS84 in meters for the new datum (input),
sigma_X	The standard error in X in meters for the new datum (input),
sigma_Y	The standard error Y in meters for the new datum (input),
sigma_Z	The standard error Z in meters for the new datum (input),

south_latitude	The southern edge of the validity rectangle for the new datum in radians (input),
north_latitude	The southern edge of the validity rectangle for the new datum in radians (input),
west_longitude	The western edge of the validity rectangle for the new datum in radians (input),
east_longitude	The eastern edge of the validity rectangle for the new datum in radians (input).

Example:

```
status = Create_Datum (code, name, ellipsoid_code,
                       delta_x, delta_y, delta_z,
                       sigma_x, sigma_Y, sigma_Z
                       south_latitude, north_latitude,
                       west_longitude, east_longitude)
```

Inputs:

code	"XXX-M"
name	"Experimental datum"
ellipsoid_code	"WGE"
delta_x	25.0
delta_y	25.0
delta_z	5.0
sigma_x	2.0
sigma_y	3.0
sigma_z	1.0
south_latitude	-1.57079632…
north_latitude	1.57079632…
west_longitude	-3.14159265…
east_longitude	3.14159265…

5.2.3 DECLARATIONS

5.2.3.1 TYPES

Not applicable.

5.2.3.2 CONSTANTS

Not applicable.

5.2.3.3 VARIABLES

Not applicable.

5.2.4 DEPENDENCIES

None.

5.2.5 ERROR HANDLING

This function returns the following status codes:

```
DATUM_NO_ERROR                  : No errors occurred in function
DATUM_NOT_INITIALIZED_ERROR     : Datum module has not been initialized
                                  properly
DATUM_INVALID_CODE_ERROR        : Datum code already found in list
DATUM_LAT_ERROR                 : Latitude out of valid range (-90 to 90)
DATUM_LON_ERROR                 : Longitude out of valid range (-180 to 360)
DATUM_SIGMA_ERROR               : Standard error values must be positive (or
                                  -1 if unknown)
DATUM_DOMAIN_ERROR              : Domain of validity not well defined
DATUM_ELLIPSE_ERROR             : Error in ellipsoid module
DATUM_3PARAM_OVERFLOW_ERROR     : 3 parameter table overflow
DATUM_3PARAM_FILE_OPEN_ERROR    : 3 parameter file opening error
```

5.3 GET_DATUM_COUNT

5.3.1 DESCRIPTION

This function returns the number of datums in the table.

5.3.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Datum_Count (long *count);
```

count Number of datums in the table (output).

5.3.3 DECLARATIONS

5.3.3.1 TYPES

Not applicable.

5.3.3.2 CONSTANTS

Not applicable.

5.3.3.3 VARIABLES

Not applicable.

5.3.4 DEPENDENCIES

None.

5.3.5 ERROR HANDLING

This function returns the following status codes:

```
DATUM_NO_ERROR                  : No errors occurred in function
DATUM_NOT_INITIALIZED_ERROR     : Datum module has not been initialized
                                    properly
```

5.4 DATUM_INDEX

5.4.1 DESCRIPTION

This function returns the index of the datum with the specified code.

5.4.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Datum_Index ( const char *code,
             long *index);
```

code The five letter code for a datum (input),

	index	The index of the datum in the table with the specified code (output).

Example:

```
status = Datum_Index(code, index)
```

Inputs:

code	"TOY-C"

Outputs:

index	201

5.4.3 DECLARATIONS

5.4.3.1 TYPES

Not applicable.

5.4.3.2 CONSTANTS

Not applicable.

5.4.3.3 VARIABLES

Not applicable.

5.4.4 DEPENDENCIES

None.

5.4.5 ERROR HANDLING

This function returns the following status codes:

```
DATUM_NO_ERROR                  : No errors occurred in function
DATUM_NOT_INITIALIZED_ERROR     : Datum module has not been initialized
                                  properly
DATUM_INVALID_CODE_ERROR        : Datum code not found in list
```

5.5 DATUM_CODE

5.5.1 DESCRIPTION

This function returns the 5-letter code of the datum referenced by index.

5.5.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Datum_Code (const long index,
                 char *code);
```

index	The index of a given datum in the datum table (input),
code	The datum code of the datum referenced by index (output).

5.5.3 DECLARATIONS

5.5.3.1 TYPES

Not applicable.

5.5.3.2 CONSTANTS

Not applicable.

5.5.3.3 VARIABLES

Not applicable.

5.5.4 DEPENDENCIES

None.

5.5.5 ERROR HANDLING

This function returns the following status codes:

```
DATUM_NO_ERROR                  : No errors occurred in function
DATUM_NOT_INITIALIZED_ERROR     : Datum module has not been initialized
                                    properly
```

```
DATUM_INVALID_INDEX_ERROR        : Index out of valid range
```

5.6 DATUM_NAME

5.6.1 DESCRIPTION

This function returns the name of the datum referenced by index.

5.6.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Datum_Name (const long index,
                 char *name);
```

index	The index of a given datum in the datum table (input),
name	The name of the datum referenced by index (output).

Example:

```
status = Datum_Name(index, name)
```

Inputs:

Index	201

Outputs:

name	"TOKYO Okinawa"

5.6.3 DECLARATIONS

5.6.3.1 TYPES

Not applicable.

5.6.3.2 CONSTANTS

Not applicable.

5.6.3.3 VARIABLES

Not applicable.

5.6.4 DEPENDENCIES

None.

5.6.5 ERROR HANDLING

This function returns the following status codes:

```
DATUM_NO_ERROR                  : No errors occurred in function
DATUM_NOT_INITIALIZED_ERROR     : Datum module has not been initialized
                                  properly
DATUM_INVALID_INDEX_ERROR       : Index out of valid range
```

5.7 DATUM_ELLIPSOID_CODE

5.7.1 DESCRIPTION

This function returns the 2-letter ellipsoid code for the ellipsoid associated with the datum referenced by index.

5.7.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Datum_Ellipsoid_Code (const long index,
             char *code);
```

index	The index of a given datum in the datum table (input),
code	The ellipsoid code for the ellipsoid associated with the datum referenced by index (output).

Function Call:

```
status = Datum_Ellipsoid_Code(index, code)
```

Inputs:

index	201

Outputs:

code	"BR"

5.7.3 DECLARATIONS

5.7.3.1 TYPES

Not applicable.

5.7.3.2 CONSTANTS

Not applicable.

5.7.3.3 VARIABLES

Not applicable.

5.7.4 DEPENDENCIES

None.

5.7.5 ERROR HANDLING

This function returns the following status codes:

```
DATUM_NO_ERROR                  : No errors occurred in function
DATUM_NOT_INITIALIZED_ERROR     : Datum module has not been initialized
                                  properly
DATUM_INVALID_INDEX_ERROR       : Index out of valid range
```

5.8 GET_DATUM_TYPE

5.8.1 DESCRIPTION

This function returns the type (3-parameter, 7-parameter, WGS 72, or WGS 84) of the datum referenced by index.

5.8.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Datum_Type (const long index,
             Datum_Type *type);
```

index The index of a given datum in the datum table (input),

type The type of the datum referenced by index (output).

Example:

```
status = Get_Datum_Type(index, type)
```

Inputs:

index 201

Outputs:

type Three_Param_Datum

5.8.3 DECLARATIONS

5.8.3.1 TYPES

Not applicable.

5.8.3.2 CONSTANTS

Not applicable.

5.8.3.3 VARIABLES

Not applicable.

5.8.4 DEPENDENCIES

None.

5.8.5 ERROR HANDLING

This function returns the following status codes:

```
DATUM_NO_ERROR                  : No errors occurred in function
DATUM_NOT_INITIALIZED_ERROR     : Datum module has not been initialized
                                  properly
DATUM_INVALID_INDEX_ERROR       : Index out of valid range
```

5.9 DATUM_SEVEN_PARAMETERS

5.9.1 DESCRIPTION

This function returns the seven parameters for the datum referenced by index.

5.9.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Datum_Seven_Parameters (const long index,
             double *delta_X,
             double *delta_Y,
             double *delta_Z,
             double *rX,
             double *rY,
             double *rZ,
             double *scale_factor);
```

index	The index of a given datum in the datum table (input),
delta_X	The X translation in meters (output),
delta_Y	The Y translation in meters (output),
delta_Z	The Z translation in meters (output),
rX	The X rotation in radians (output),
rY	The Y rotation in radians (output),
rZ	The Z rotation in radians (output),
scale_factor	The scale factor (output).

5.9.3 DECLARATIONS

5.9.3.1 TYPES

Not applicable.

5.9.3.2 CONSTANTS

Not applicable.

5.9.3.3 VARIABLES

Not applicable.

5.9.4 DEPENDENCIES

None.

5.9.5 ERROR HANDLING

This function returns the following status codes:

```
DATUM_NO_ERROR                  : No errors occurred in function
DATUM_NOT_INITIALIZED_ERROR     : Datum module has not been initialized
                                  properly
DATUM_INVALID_INDEX_ERROR       : Index out of valid range
```

5.10 DATUM_THREE_PARAMETERS

5.10.1 DESCRIPTION

This function returns the three parameters for the datum referenced by index.

5.10.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Datum_Three_Parameters (const long index,
             double *delta_X,
             double *delta_Y,
             double *delta_Z);
```

index	The index of a given datum in the datum table (input),
delta_X	The X translation in meters (output),
delta_Y	The Y translation in meters (output),
delta_Z	The Z translation in meters (output).

Example:

```
status = Datum_Three_Parameters(index, delta_X, delta_Y, delta_Z)
```

Inputs:

index:	201

Outputs:

delta_X	-158
delta_Y	507
delta_Z	676

5.10.3 DECLARATIONS

5.10.3.1 TYPES

Not applicable.

5.10.3.2 CONSTANTS

Not applicable.

5.10.3.3 VARIABLES

Not applicable.

5.10.4 DEPENDENCIES

None.

5.10.5 ERROR HANDLING

This function returns the following status codes:

```
DATUM_NO_ERROR                  : No errors occurred in function
DATUM_NOT_INITIALIZED_ERROR     : Datum module has not been initialized
                                   properly
DATUM_INVALID_INDEX_ERROR       : Index out of valid range
```

5.11 DATUM_ERRORS

5.11.1 DESCRIPTION

This function returns the standard errors in X, Y, Z for the datum referenced by index.

5.11.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Datum_Errors (const long index,
             double *sigma_X,
             double *sigma_Y,
             double *sigma_Z);
```

index	The index of a given datum in the datum table (input),
sigma_X	The standard error in X in meters (output),
sigma_Y	The standard error Y in meters (output),
sigma_Z	The standard error Z in meters (output).

Example:

```
status = Datum_Errors(index, delta_X, delta_Y, delta_Z)
```

Inputs:

index:	201

Outputs:

sigma _X	20.0
sigma _Y	5.0
sigma _Z	20.0

5.11.3 DECLARATIONS

5.11.3.1 TYPES

Not applicable.

5.11.3.2 CONSTANTS

Not applicable.

5.11.3.3 VARIABLES

Not applicable.

5.11.4 DEPENDENCIES

None.

5.11.5 ERROR HANDLING

This function returns the following status codes:

```
DATUM_NO_ERROR                  : No errors occurred in function
DATUM_NOT_INITIALIZED_ERROR     : Datum module has not been initialized
                                  properly
DATUM_INVALID_INDEX_ERROR       : Index out of valid range
```

5.12 DATUM_VALID_RECTANGLE

5.12.1 DESCRIPTION

This function returns the edges of the validity rectangle for the datum referenced by index.

5.12.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Datum_Valid_Rectangle (const long index,
            double *south_latitude,
            double *north_latitude,
            double *west_longitude,
            double *east_longitude);
```

index	The index of a given datum in the datum table (input),
south_latitude	The southern edge of the validity rectangle for the datum in radians (output),
north_latitude	The southern edge of the validity rectangle for the datum in radians (output),
west_longitude	The western edge of the validity rectangle for the datum in radians (output),
east_longitude	The eastern edge of the validity rectangle for the datum in radians (output).

Example:

```
status = Datum_Valid_Rectangle (index, south_latitude, north_latitude,
                        west_longitude, east_longitude)
```

Inputs:

index:	201

Outputs:

south_latitude	19.0 * TWO_PI/180.0
north_latitude	31.0 * TWO_PI/180.0
west_longitude	119.0 * TWO_PI/180.0
east_longitude	134.0 * TWO_PI/180.0

5.12.3 DECLARATIONS

5.12.3.1 TYPES

Not applicable.

5.12.3.2 CONSTANTS

Not applicable.

5.12.3.3 VARIABLES

Not applicable.

5.12.4 DEPENDENCIES

None.

5.12.5 ERROR HANDLING

This function returns the following status codes:

```
DATUM_NO_ERROR                  : No errors occurred in function
DATUM_NOT_INITIALIZED_ERROR     : Datum module has not been initialized
                                  properly
DATUM_INVALID_INDEX_ERROR       : Index out of valid range
```

5.13 VALID_DATUM

5.13.1 DESCRIPTION

This function checks whether or not the specified location is within the validity rectangle for the specified datum. It returns.zero if the specified location is NOT within the validity rectangle, and returns 1 otherwise.

5.13.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Valid_Datum (const long index,
             double latitude,
             double longitude,
             long *result);
```

index	The index of a given datum in the datum table (input),
latitude	The latitude of the specified location in radians (input),
longitude	The longitude of the specified location in radians (input),
result	Indicates whether the specified location is within (1) or outside of (0) the validity rectangle for the specified datum (output).

Example:

```
status = Valid Datum (index, latitude, longitude, result);
```

Inputs:

index:	201
latitude	25.0 * TWO_PI/180.0
longitude	126.0 * TWO_PI/180.0

Outputs:

result	1

5.13.3 DECLARATIONS

5.13.3.1 TYPES

Not applicable.

5.13.3.2 CONSTANTS

Not applicable.

5.13.3.3 VARIABLES

Not applicable.

5.13.4 DEPENDENCIES

None.

5.13.5 ERROR HANDLING

This function returns the following status codes:

```
DATUM_NO_ERROR                  : No errors occurred in function
DATUM_NOT_INITIALIZED_ERROR     : Datum module has not been initialized
                                  properly
DATUM_INVALID_INDEX_ERROR       : Index out of valid range
DATUM_LAT_ERROR                 : Latitude out of valid range (-90 to 90)
DATUM_LON_ERROR                 : Longitude out of valid range (-180 to 360)
```

5.14 GEOCENTRIC_DATUM_SHIFT

5.14.1 DESCRIPTION

This function shifts a geocentric coordinate (X, Y, Z in meters) relative to the source datum to geocentric coordinate (X, Y, Z in meters) relative to the destination datum.

5.14.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Geocentric_Datum_Shift (const long index_in,
             const double X_in,
             const double Y_in,
             const double Z_in,
             const long index_out,
             double X_out,
             double Y_out,
             double Z_out);
```

index_in The index of the source datum in the datum table (input),

X_in The X coordinate relative to the source datum in meters (input),

Y_in	The Y coordinate relative to the source datum in meters (input),
Z_in	The Z coordinate relative to the source datum in meters (input),
index_out	The index of the destination datum in the datum table (input),
X_out	The X coordinate relative to the destination datum in meters (output),
Y_out	The Y coordinate relative to the destination datum in meters (output),
Z_out	The Z coordinate relative to the destination datum in meters (output).

5.14.3 DECLARATIONS

5.14.3.1 TYPES

Not applicable.

5.14.3.2 CONSTANTS

Not applicable.

5.14.3.3 VARIABLES

Not applicable.

5.14.4 DEPENDENCIES

Geocentric_Shift_To_WGS84 – used to perform a datum shift from a specified local datum to WGS 84, and

Geocentric_Shift_From_WGS84 – used to perform a datum shift from WGS 84 to a specified local datum.

5.14.5 ERROR HANDLING

This function returns the following status codes:

```
DATUM_NO_ERROR                  : No errors occurred in function
DATUM_NOT_INITIALIZED_ERROR     : Datum module has not been initialized
                                  properly
DATUM_INVALID_SOURCE_INDEX_ERROR : Source datum index invalid
```

```
DATUM_INVALID_DEST_INDEX_ERROR   : Destination datum index invalid
```

5.15 GEOCENTRIC_SHIFT_TO_WGS84

5.15.1 DESCRIPTION

This function shifts geocentric coordinates (X, Y, Z in meters) relative to the datum referenced by index to geocentric coordinates (X, Y, Z in meters) relative to WGS 84.

5. 15.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Geocentric_Shift_To_WGS84 (const long index,
            const double X,
            const double Y,
            const double Z,
            double *X_WGS84,
            double *Y_WGS84,
            double *Z_WGS84);
```

index	The index of the source datum in the datum table (input),
X	The X coordinate relative to the source datum in meters (input),
Y	The Y coordinate relative to the source datum in meters (input),
Z	The Z coordinate relative to the source datum in meters (input),
X_WGS84	The X coordinate relative to WGS 84 in meters (output),
Y_WGS84	The Y coordinate relative to WGS 84 in meters (output),
Z_WGS84	The Z coordinate relative to WGS 84 in meters (output).

Example:

```
status = Geocentric_Shift_To_WGS84 (index, X, Y, Z, X_WGS84, Y_WGS84, Z_WGS84)
```

Inputs:

index	201
X	0.0
Y	0.0
Z	0.0

Outputs:

X_WGS84	-158.0

Y_WGS84 507.0

Z_WGS84 676.0

5. 15.3 DECLARATIONS

5. 15.3.1 TYPES

Not applicable.

5. 15.3.2 CONSTANTS

Not applicable.

5. 15.3.3 VARIABLES

Not applicable.

5. 15.4 DEPENDENCIES

Geocentric_Shift_WGS72_To_WGS84 – used to perform a datum shift from WGS 72 to WGS 84.

5. 15.5 ERROR HANDLING

This function returns the following status codes:

```
DATUM_NO_ERROR                  : No errors occurred in function
DATUM_NOT_INITIALIZED_ERROR     : Datum module has not been initialized
                                    properly
DATUM_INVALID_SOURCE_INDEX_ERROR : Source datum index invalid
```

5.16 GEOCENTRIC_SHIFT_FROM_WGS84

5.16.1 DESCRIPTION

This function shifts geocentric coordinates (X, Y, Z in meters) relative to WGS 84 to geocentric coordinates (X, Y, Z in meters) relative to the datum referenced by index.

5.16.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Geocentric_Shift_From_WGS84 (const double X_WGS84
            const double Y_WGS84,
            const double Z_WGS84,
            const long index,
            double *X,
            double *Y,
            double *Z);
```

X_WGS84	The X coordinate relative to WGS 84 in meters (input),
Y_WGS84	The Y coordinate relative to WGS 84 in meters (input),
Z_WGS84	The Z coordinate relative to WGS 84 in meters (input),
index	The index of the destination datum in the datum table (input),
X	The X coordinate relative to the destination datum in meters (output),
Y	The Y coordinate relative to the destination datum in meters (output),
Z	The Z coordinate relative to the destination datum in meters (output).

5.16.3 DECLARATIONS

5.16.3.1 TYPES

Not applicable.

5.16.3.2 CONSTANTS

Not applicable.

5.16.3.3 VARIABLES

Not applicable.

5.16.4 DEPENDENCIES

Geocentric_Shift_WGS84_To_WGS72 – used to perform a datum shift from WGS 84 to WGS 72.

5.16.5 ERROR HANDLING

This function returns the following status codes:

```
DATUM_NO_ERROR                  : No errors occurred in function
DATUM_NOT_INITIALIZED_ERROR     : Datum module has not been initialized
                                    properly
DATUM_INVALID_DEST_INDEX_ERROR  : Destination datum index invalid
```

5.17 GEODETIC_DATUM_SHIFT

5. 17.1 DESCRIPTION

This function shifts geodetic coordinates (latitude, longitude in radians and height in meters) relative to the source datum to geodetic coordinate (latitude, longitude in radians and height in meters) relative to the destination datum.

5. 17.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Geodetic_Datum_Shift (const long index_in,
            const double Lat_in,
            const double Lon_in,
            const double Hgt_in,
            const long index_out,
            double Lat_out,
            double Lon_out,
            double Hgt_out);
```

index_in	The index of the source datum in the datum table (input),
Lat_in	The latitude relative to the source datum in radians (input),
Lon_in	The longitude relative to the source datum in radians (input),
Hgt_in	The height relative to the source datum in meters (input),
index_in	The index of the destination datum in the datum table (input),
Lat_out	The latitude relative to the destination datum in radians (output),
Lon_out	The longitude relative to the destination datum in radians (output),
Hgt_out	The height relative to the destination datum in meters (output).

5. 17.3 DECLARATIONS

5. 17.3.1 TYPES

Not applicable.

5. 17.3.2 CONSTANTS

Not applicable.

5. 17.3.3 VARIABLES

Not applicable.

5. 17.4 DEPENDENCIES

Geodetic_Shift_To_WGS84 – used to perform a datum shift in geodetic coordinates from a specified local datum to WGS 84, and

Geodetic_Shift_From_WGS84 – used to perform a datum shift in geodetic coordinates from WGS 84 to a specified local datum.

5. 17.5 ERROR HANDLING

This function returns the following status codes:

```
DATUM_NO_ERROR                   : No errors occurred in function
DATUM_NOT_INITIALIZED_ERROR      : Datum module has not been initialized
                                    properly
DATUM_INVALID_SOURCE_INDEX_ERROR : Source datum index invalid
DATUM_INVALID_DEST_INDEX_ERROR   : Destination datum index invalid
DATUM_LAT_ERROR                  : Latitude out of valid range (-90 to 90)
DATUM_LON_ERROR                  : Longitude out of valid range (-180 to 360)
DATUM_ELLIP_ERROR                : Can't get ellipsoid information for
                                    ellipsoid that is associated with datum
                                    index.
```

5.18 GEODETIC_SHIFT_TO_WGS84

5. 18.1 DESCRIPTION

This function shifts geodetic coordinates (latitude, longitude in radians and height in meters) relative to the datum referenced by index to geodetic coordinate (latitude, longitude in radians and height in meters) relative to WGS 84.

5. 18.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Geodetic_Shift_To_WGS84 (const long Index,
             const double Lat_in,
             const double Lon_in,
             const double Hgt_in,
             double *WGS84_Lat,
             double *WGS84_Lon,
             double *WGS84_Hgt);
```

Index	The index of the source datum in the datum table (input),
Lat_in	The latitude relative to the source datum in radians (input),
Lon_in	The longitude relative to the source datum in radians (input),
Hgt_in	The height relative to the source datum in meters (input),
WGS84_Lat	The latitude relative to WGS 84 in radians (output),
WGS84_Lon	The longitude relative to WGS 84 in radians (output),
WGS84_Hgt	The height relative to WGS 84 in meters (output).

5. 18.3 DECLARATIONS

5. 18.3.1 TYPES

Not applicable.

5. 18.3.2 CONSTANTS

Not applicable.

5. 18.3.3 VARIABLES

Not applicable.

5. 18.4 DEPENDENCIES

Geodetic_Shift_WGS72_To_WGS84 – used to perform a datum shift in geodetic coordinates from WGS 72 to WGS 84,

Molodensky_Shift – used to perform a datum shift from one datum to another using Molodensky's method,

Get_Ellipsoid_Index from ELLIPSOID component – used to look up the ellipsoid table index corresponding a particular ellipsoid code, and

Get_Ellipsoid_Axes from ELLIPSOID component – used to obtain the lengths of the semi-major and semi-major axes, in meters, of the specified.

5. 18.5 ERROR HANDLING

This function returns the following status codes:

```
DATUM_NO_ERROR                   : No errors occurred in function
DATUM_NOT_INITIALIZED_ERROR      : Datum module has not been initialized
                                   properly
DATUM_INVALID_SOURCE_INDEX_ERROR : Source datum index invalid
DATUM_LAT_ERROR                  : Latitude out of valid range (-90 to 90)
DATUM_LON_ERROR                  : Longitude out of valid range (-180 to 360)
DATUM_ELLIP_ERROR                : Can't get ellipsoid information for
                                   ellipsoid that is associated with datum
                                   index.
```

5.19 GEODETIC_SHIFT_FROM_WGS84

5. 19.1 DESCRIPTION

This function shifts geodetic coordinates (latitude, longitude in radians and height in meters) relative to WGS 84 to geodetic coordinate (latitude, longitude in radians and height in meters) relative to the datum referenced by index.

5. 19.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Geodetic_Shift_From_WGS84 (const double WGS84_Lat
```

```
        const double WGS84_Lon,
        const double WGS84_Hgt,
        const long index,
        double *Lat_out,
        double *Lon_out,
        double *Hgt_out);
```

WGS84_Lat	The latitude relative to WGS 84 in radians (input),
WGS84_Lon	The longitude relative to WGS 84 in radians (input),
WGS84_Hgt	The height relative to WGS 84 in meters (input),
index	The index of the destination datum in the datum table (input),
Lat_out	The latitude relative to the destination datum in radians (output),
Lon_out	The longitude relative to the destination datum in radians (output),
Hgt_out	The height relative to the destination datum in meters (output).

5. 19.3 DECLARATIONS

5. 19.3.1 TYPES

Not applicable.

5. 19.3.2 CONSTANTS

Not applicable.

5. 19.3.3 VARIABLES

Not applicable.

5. 19.4 DEPENDENCIES

Geodetic_Shift_WGS84_To_WGS72 – used to perform a datum shift in geodetic coordinates from WGS 84 to WGS 72,

Molodensky_Shift – used to perform a datum shift from one datum to another using Molodensky's method,

Get_Ellipsoid_Index from ELLIPSOID component – used to look up the ellipsoid table index corresponding a particular ellipsoid code, and

Get_Ellipsoid_Axes from ELLIPSOID component – used to obtain the lengths of the semi-major and semi-major axes, in meters, of the specified.

5. 19.5 ERROR HANDLING

This function returns the following status codes:

```
DATUM_NO_ERROR                    : No errors occurred in function
DATUM_NOT_INITIALIZED_ERROR       : Datum module has not been initialized
                                    properly
DATUM_INVALID_DEST_INDEX_ERROR    : Destination datum index invalid
DATUM_LAT_ERROR                   : Latitude out of valid range (-90 to 90)
DATUM_LON_ERROR                   : Longitude out of valid range (-180 to 360)
DATUM_ELLIP_ERROR                 : Can't get ellipsoid information for
                                    ellipsoid that is associated with datum
                                    index.
```

5.20 DATUM_SHIFT_ERROR

5. 20.1 DESCRIPTION

This function returns the 90% horizontal (circular), vertical (linear), and spherical errors for a shift from the specified source datum to the specified destination datum at the specified location.

5. 20.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Datum_Shift_Error (const long index_in,
             const long index_out,
             double latitude,
             double longitude,,
             double *ce90,
             double *le90,
             double *se90);
```

index_in	The index of the source datum (input),
index_out	The index of the destination datum (input),
latitude	The latitude of the point being shifted (input),
longitude	The longitude of the point being shifted (input),
ce90	Combined 90% circular horizontal error in meters (output),
le90	Combined 90% linear vertical error in meters (output),
se90	Combined 90% spherical error in meters (output).

5. 20.3 DECLARATIONS

5. 20.3.1 TYPES

Not applicable.

5. 20.3.2 CONSTANTS

Not applicable.

5. 20.3.3 VARIABLES

Not applicable.

5. 20.4 DEPENDENCIES

Not applicable.

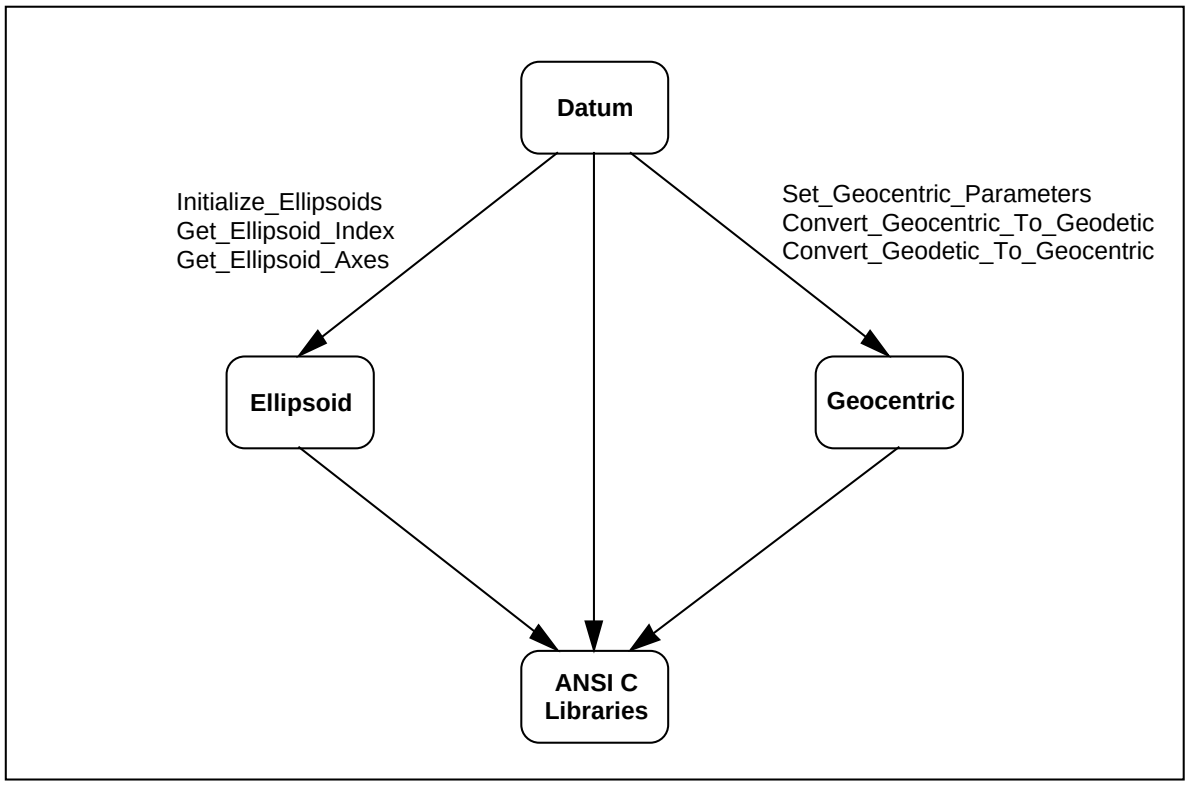
5. 20.5 ERROR HANDLING

This function returns the following status codes:

```
DATUM_NO_ERROR                  : No errors occurred in function
DATUM_NOT_INITIALIZED_ERROR     : Datum module has not been initialized
                                  properly
DATUM_INVALID_SRC_INDEX_ERROR   : Source datum index invalid
DATUM_INVALID_DEST_INDEX_ERROR  : Destination datum index invalid
DATUM_LAT_ERROR                 : Latitude out of valid range (-90 to 90)
DATUM_LON_ERROR                 : Longitude out of valid range (-180 to 360)
```

APPENDIX A STRUCTURE/DEPENDENCY DIAGRAMS

This component consists of a single compilation unit and depends on the ANSI C standard character handling, math, input/output, string handling, and general utility libraries, as well as on the ELLIPSOID and GEOCENTRIC components.



APPENDIX B DEFINITIONS/GLOSSARY

Datum – Any numerical or geometrical quantity or set of such quantities specifying the reference coordinate system used for geodetic control in the calculation of coordinates of points on the earth. Datums may be either global or local in extent. A local datum defines a coordinate system that is used only over a region of limited extent. A global datum specifies the center of the reference ellipsoid to be located at the earth's center of mass and defines a coordinate system used for the entire earth.

Ellipsoid – The surface generated by an ellipse rotating about one of its axes.

Geocentric Coordintes – Cartesian coordinates (X, Y, Z) that define the position of a point with respect to the center of mass of the earth.

Geodetic Coordinates – The quantities of latitude and longitude that define the position of a point on the surface of the earth with respect to the reference ellipsoid. Also, imprecisely called geographic coordinates.

Geodetic Height – The height above the reference ellipsoid, measured along the ellipsoidal normal through the point in question. The geodetic height is positive if the point is outside the ellipsoid. Also known as ellipsoidal height, h.

Geodetic Latitude – The angle between the plane of the equator and the normal to the ellipsoid through the computation point. Geodetic latitude is positive north of the equator and negative south of the equator.

Geodetic Longitude – The angle between the plane of a meridian and the plane of the prime meridian. A longitude can be measured from the angle formed between the local and prime meridians at the pole of rotation of the reference ellipsoid, or by the arc along the equator intercepted by these meridians.

Geoid – The equipotential surface of the earth's gravity field approximated by undisturbed mean sea level fo the oceans. The direction of gravity passing through a given point on the geoid is perpendicular to this equipotential surface.

Geoid Separation – The distance between the geoid and the mathematical reference ellipsoid as measured along the ellipsoidal normal. This distance is positive outside, or negative inside, the reference ellipsoid. Also called geoidal height; undulation of the geoid.

Horizontal Datum – A horizontal datum specifies the coordinate system in which latitude and longitude of points are located. The latitude and longitude of an initial point, the azimuth of a line from that point, and the semi-major axis and flattening of the ellipsoid that approximates the surface of the earth in the region of interest define a horizontal datum.

Reference Ellipsoid – An ellipsoid whose dimensions closely approach the dimensions of the geoid; the exact dimensions are determined by various considerations of the section of the earth's surface concerned. Usually a bi-axial ellipsoid of revolution.

Vertical Datum – A vertical datum is the surface to which elevations are referred. A local vertical datum is a continuous surface, usually mean sea level, at which elevations are assumed to be zero throughout the area of interest.

WGS 72 – World Geodetic System 1972. WGS 72 was the previous DoD standard earth-centered, earth-fixed world geodetic system. It was superceded by WGS 84

WGS 84 – World Geodetic System 1984. A world geodetic system provides the basic reference frame, geometric figure and gravimetric model for the earth, and provides the means for relating positions on various local geodetic systems to an earth-centered, earth-fixed coordinate system. WGS 84 is the current DoD standard earth-centered, earth-fixed world geodetic system.

APPENDIX C REFERENCES

(1) Topographic Engineering Center, TEC-SR-7, **Handbook for transformation of DATUMS, PROJECTIONS, GRIDS, AND COMMON COORDINATE SYSTEMS**, January 1996.