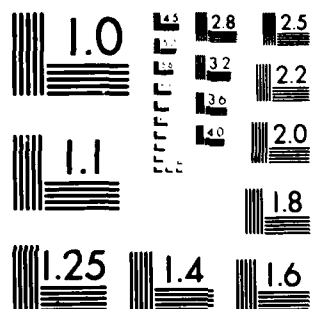


US ARMY INTELLIGENCE CENTER AND SCHOOL USAICS SOFTWARE
ANALYSIS AND MANAG. (U) JET PROPULSION LAB PASADENA CA
P BABBY ET AL. 09 JUL 82 ~~762-181~~ NAS7-918

F/G 8/2. NL

~~END
PAGE
FIXED
7-83
DTIC~~

Cont



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

AD A129182

UNCLAS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER UAA002	2. GOVT ACCESSION NO. AD 4129 102	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) U.S. ARMY INTELLIGENCE CENTER AND SCHOOL USAICS, Software Analysis and Management System Analysis of Geographic Transformation Algorithms.		5. TYPE OF REPORT & PERIOD COVERED FINAL
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) JET PROPULSION LABORATORY, California Institute of Technology, Pasadena, California		8. CONTRACT OR GRANT NUMBER(s) NAS7- 918
9. PERFORMING ORGANIZATION NAME AND ADDRESS Jet Propulsion Laboratory ATTN: 171-209 California Institute of Technology 4800 Oak Grove, Pasadena, CA 91109		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS RE 182 AMEND # 187
11. CONTROLLING OFFICE NAME AND ADDRESS Commander USAICS ATTN: ATSI-CD-SF Ft Huachuca, AZ 85613		12. REPORT DATE JULY 9, 1982
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (Same as ITEM 11) Commander, USAICS.....		13. NUMBER OF PAGES 35
		15. SECURITY CLASS. (of this report) UNCLAS
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE NONE		
16. DISTRIBUTION STATEMENT (of this Report) Approved for Public dissemination		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES Prepared by the Jet Propulsion Labs for the US Army Intelligence Center and School.		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) GEOGRAPHIC TRANSFORMATI(ALGORITHM.		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report describes the findings of JPL regarding geographic transformation algorithms used in MAGIIC, Guardrail, Trailblazer and BETA systems. A set of parameters is developed to characterize and catalogue intelligence system algo- rithms in the 4 systems. Individual algorithms are also analyzed to determine if they are performing their functions properly.		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 68 IS OBSOLETE

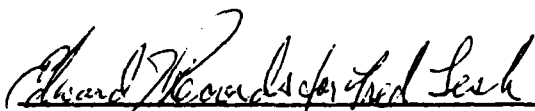
UNCLAS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

U.S. ARMY INTELLIGENCE CENTER AND SCHOOL
USAICS
Software Analysis and Management System
Analysis of Geographic Transformation Algorithms

July 9, 1982

Approval:



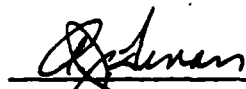
H. F. Lesh, Manager
USAICS Software Analysis and
Management System



John Radbill, Task Leader
Algorithm Analysis
USAICS Software Analysis and
Management System



H. Jeanne, Manager
Software Development Section



A. Ferrari, Manager
Defense Information Systems Program

Concur:



Major W. Akins, Chief
All Source Analysis System Office
U.S. Army Intelligence Center and School

DTIC
ELECTE
S JUN 9 1983 D

This document has been approved
for public release and sale; its
distribution is unlimited.

JET PROPULSION LABORATORY
California Institute of Technology
Pasadena, California

The following people contributed to this report

P. Babby
J. Gillis
A. Griesel
N. Palmer
J. Radbill

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By _____	
Distribution/	
Availability Codes	
Avail and/or	
Disc	Special



TABLE OF CONTENTS

1. Introduction	1
1.1. Purpose.....	1
1.2. Background.....	1
1.3. User Benefits.....	3
2. Analyzing the Algorithms.....	4
2.1. Early Steps.....	4
2.2. Learning Military Mapping.....	4
2.3. Representing Algorithms in Standard Form.....	7
3. Characterizing and Cataloging the Algorithms.....	12
4. Algorithms in MAGIIC.....	14
4.1. MAGIIC Lambert Constant Generation Algorithm.....	14
4.2. An Interrelated Set of MAGIIC Coordinate Conversion Algorithms.....	14
4.3. MAGIIC Polar Grid to Universal Polar Sterographic Algorithm.....	16
4.4. MAGIIC Northing and Easting to Universal Transverse Mercator Algorithm..	17
5. Comparing the Algorithms Across Systems.....	19
5.1. Spheroid Models.....	19
5.2. Grid Zone Generation.....	19
5.3. MAGIIC Grid Zone Generation Algorithm.....	20
5.4. Guardrail Grid Zone Generation Algorithm.....	20
5.5. Trailblazer Grid Zone Generation Algorithm.....	21
5.6. BETA Grid Zone Generation Algorithm.....	21
6. Discussion and Conclusions.....	24
6.1. Documentation of Algorithms.....	24
6.2. Similarity of Functions Across Systems.....	24
6.3. Incompleteness of Algorithms for Global Applications.....	24
6.4. Robustness of Algorithms.....	24
6.5. Selection/Consolidation of Algorithms.....	25

7. Appendices.....	27
7.1. Database Entries and Products (PSL/PSA).....	28
7.2. Algorithm Hierarchy Charts.....	29
7.3. Algorithms in Standard Form.....	30

LIST OF FIGURES

2-1: Grid Zone Designations of the Military Grid Reference System (UTM).....	9
2-2: Grid Zone Designations of Universal Polar Stereographic (UPS).....	10
2-3: Basic Plan of the 100,000-meter Square Identifications	11
3-1: Algorithm Level Help File.....	13
6-1: Documentation of Algorithms.....	26

LIST OF TABLES

5-1: Inconsistent Spheroid Usage.....	23
---------------------------------------	----

1. Introduction

1.1. Purpose

This report^a describes the findings of the Algorithm Analysis Subtask group working on the U.S. Army Intelligence Center and School (USAICS) Software Analysis and Management System task (USAMS) regarding geographic transformation algorithms used in four of the intelligence-gathering systems under USAICS cognizance. In this report a set of parameters is developed to characterize and catalogue intelligence system algorithms in four specific systems. Individual algorithms are analyzed to determine whether they are performing their functions properly. Algorithms that perform the same function in different systems are compared to determine which ones are best according to various criteria.

The algorithms examined in this report are taken from the MAGIIC, Guardrail, Trailblazer, and BETA systems. They were chosen from the approximately 41 deployed intelligence systems for which USAICS is Combat Developer because their documentation was quickly accessible and because they represented a range of algorithm applications. Geographic transformation (mapping) algorithms were chosen for this report since all four systems contain position location/description functions and many of their algorithms are unclassified.

1.2. Background

Each of the about 41 intelligence systems under USAICS cognizance employs several types of algorithms to carry out its gathering and processing of intelligence data. Two important types of these algorithms, geographic transformation and correlation, have been chosen for analysis during this year. The former translates grid zone locations, for example, from latitude-longitude to Universal Transverse Mercator (UTM), while the latter resolves many individual sitings into militarily recognizable targets based chiefly on standard statistical procedures. It is important to develop a set of parameters to characterize these algorithms so that how they should be catalogued can be

^aTwo additional reports will be submitted in FY82: a correlation analysis report and a report on possible algorithm analysis methodologies.

determined. When these activities are completed, it becomes possible to compare algorithms that perform the same function in different systems.

To begin this process, the JPL Algorithm Analysis Sub-task group has examined the geographic transformation algorithms for four of the 41 systems, namely The Mobile Army Ground Imagery Interpretation Center (MAGIIC), Guardrail, Trailblazer, and Battlefield Exploitation and Target Acquisition (BETA). These four systems chosen for more detailed study represent several intelligence data analysis functions. MAGIIC is a ground-based analysis system to assist in interpreting hard-copy images from different airborne surveillance systems, including a capability for computerized mensuration on imagery; it can also receive and analyze data from Tactical Electronic Intelligence (TEREC) collection systems and provide emitter location estimates. Guardrail uses airborne sensor platforms to collect data on Direction Finding (DF) emitters; extensive ground-based software is then used to estimate the location of the units, such as command posts, associated with these emitters. Trailblazer also uses DF data to estimate emitter location. Its sensor platforms are essentially fixed and ground-based. BETA is a Test Bed program for correlating data received from several types of sensor systems and making target nominations. Both automatic correlation and aggregation techniques and interactive graphics are used in the operator's analysis. These systems would generally be employed at Division or Corps level or at an Air Force Tactical Air Control Element (TACE) or Allied Tactical Air Force (ATAF); target nominations and tactical situation reports would be available to commanders and their staffs from Brigade through Echelons Above Corps (EAC).

USAICS has cognizance of a large number of algorithms integral to intelligence-gathering systems in various stages of development and deployment. The state of "deployment" of algorithms in the USAICS inventory ranges from that of products of research contracts not yet implemented in any system to those in fielded systems such as Trailblazer or Guardrail. In the latter systems the algorithms are documented in design documents (narrative English and equations), and/or in machine readable design language, and in code. Often not all of these forms of documentation are available for any one system. For research algorithms not yet implemented actual code, or even detailed flow charts, may not be available; thus analysis must rely solely on mathematical descriptions.

"Algorithm" means any set of rules for carrying out a single conceptual operation on a set of data, such as transforming latitude-longitude coordinates to UTM or determining a position from a number of direction measurements taken at known points.⁸⁸ Algorithms are often hierarchical, lower-level algorithms often being used to describe higher-level algorithms and thereby illuminating their underlying logical structure. Thus, results from one algorithm may be data for another. USAICS is interested in algorithms performing intelligence data processing functions central to their systems' mission and those performing crucial support functions, such as geographic location, common to a number of systems. Data management or mathematical function algorithms, although vital to the efficient functioning of the systems, are not being treated in these first algorithm analyses.

1.3. User Benefits

These analyses can benefit users in several ways. First, a catalog of existing algorithms will help USAICS avoid having algorithms redeveloped for new systems from first principles. Second, analysis of individual algorithms may, in a few cases, identify deficiencies worth correcting on the next system revision. Third, and most important, the comparison of algorithms performing the same function in different systems can lead to identifying guidelines for developing and/or selecting algorithms to include in new and revised systems. Selected algorithms from the systems studied will begin to form a library of intelligence algorithms with associated computer subroutines that will be analogous to the Collected Algorithms of the Association for Computing Machinery (ACM). The creation of such a library is in the spirit of Ada⁺, the Department of Defense language for embedded systems, and Ada's environment.

⁸⁸These conceptual models should be describable, although their technical implementation is often significantly more complicated to present.

⁺Ada is a trademark of the Department of Defense

2. Analyzing the Algorithms

2.1. Early Steps

Since the Location and Movement Analysis System (LAMAS) system documentation was available first, our early analysis efforts were directed to that system. A preliminary analysis of a Shortest Path Algorithm was done and modeled in Pascal as an approach to standardizing representations. This algorithm was a variant of Dijkstra's Shortest Path Algorithm.

Later the sponsor decided that our first emphasis should be on the coordinate conversion algorithms of the MAGIIC, Guardrail, Trailblazer and BETA systems. Three approaches to this analysis were tried and evaluated. The MAGIIC system has been hierarchically analyzed for the interrelationship of the algorithms. The spheroid models of the Earth's oblateness have been examined for all the systems. The grid zone generation algorithms have been compared across all four systems.

2.2. Learning Military Mapping

To analyze the first type of algorithm required learning the military grid system. This discussion identifies the various map projections and military grid reference systems examined and how they are interrelated. The scope of this discussion is limited to only those map projections and grid reference systems pertinent to the MAGIIC, BETA, Guardrail, and Trailblazer systems.

The map projections discussed are the Transverse Mercator, Polar Stereographic, Lambert Conformal Conic, and the Gnomonic. The grid reference systems used are the Universal Transverse Mercator (UTM), Universal Polar Stereographic (UPS), and Military Grid Reference System (MGR). The selection of a map projection is based on the properties it preserves in the transformation from a three-dimensional spheroid to a two-dimensional plane. These properties include orthogonality of latitude and longitude, equal area representation, distortion of shape, minimal change in scale factor in either east-west or north-south directions, and representation of great circles by straight lines. Since all map projections are from a spheroid model of the Earth, the parameters that the spheroid model use are very important. Various spheroid models are used for different portions of the Earth.

The selection of a grid reference system depends on the portion of the Earth examined and the resolution desired. The UTM and UPS coordinate systems were adopted as standards by the military to minimize coordination problems due to the proliferation of locally-used grid reference systems. These coordinate systems are most suitable for the representation of large geographic areas (greater than 9° in latitude and longitude). The MGR system provides greater resolution when representing smaller geographic areas (within 100,000-meter by 100,000-meter squares). The MGR system can be overlaid on the UTM and UPS coordinate systems to eliminate ambiguity due to repetitions of the 100,000-meter square identifiers. The geographic reference system is simply given as a longitude-latitude pair. However, this reference system of zones is cumbersome for representing locations in good resolution. Also, there is an inconsistency in the form of the coordinates: some applications use decimal degree notation while others use clock-like representations.

The UTM grid reference system is valid for all longitudes over latitudes between 84° North and 80° South. This area is divided into rectangles of 6° in longitude (zones) by 8° in latitude (bands), except for the 12° band from 72° to 84° latitude. There are 60 zones numbered from 1 through 60 for the zones from -180° to $+180^\circ$ longitude. There are 22 bands lettered C through X for the bands from -80° to 84° latitude. There are some subtle irregularities in this pattern beyond 56° latitude between 0° and 45° in longitude (see Figure 2-1). The UTM grid reference system is based on the Modified Transverse Mercator projection, but can be mathematically transformed for use with other types of projections.

The UPS grid reference system is valid for the North Polar ($+84^\circ$ longitude to the pole) and the South Polar regions (-80° longitude to the pole). These regions have a grid zone number of zero and consist only of a grid zone letter that is longitude-dependent. The North Polar region grid zone letters are Y (Western hemisphere) and Z (Eastern Hemisphere). The South Polar regions are A and B. The UPS grid reference system is based on the Polar Stereographic projection (see Figure 2-2).

The MGR grid reference system, illustrated for the UPS system in Figure 2-2 and for the UTM system in Figure 2-3, provides finer resolution than the UTM or UPS grid reference systems. It identifies 100,000-meter by 100,000-

meter squares by two letters, an Easting letter and a Northing letter. These letters are sequenced so as to provide at least 18° separation between similarly-lettered squares (within a given spheroid model area - otherwise the separation is 9°). These lettering sequences are biased and restarted at the boundaries of the underlying spheroid models.

These MGR system letter designations may be used without reference to the UTM or UPS designations when there is no likelihood of ambiguity, otherwise the UTM or UPS designation is included. Positions within these squares can be interpolated in tens of meters and are referred to as Easting and Northing terms representing distances rather than degrees.

The Transverse Mercator projection transforms the Earth's spheroid onto a cylinder secant to the Earth and perpendicular to its axis. This projection is used at latitudes within 84° North and 80° South. Scale linearity is correct at the two meridians cut by the cylinder (6° apart) and quite accurate in the band formed by them. Because of the vertical linearity this projection is particularly suitable to areas of interest in the North-South direction. This projection lends itself well to being overlaid with a rectangular grid reference system (such as the MGR system).

The Polar Stereographic projection transforms the Earth's spheroid onto a plane tangential to the Earth (at the pole, in our applications). This projection is used at latitudes beyond 84° North and from 80° South. Scale linearity decreases and equal area exaggeration increases as the distance from the pole increases. Latitude-longitude orthogonality is preserved at the meridian crossings. All circles of latitude are concentric, centered at the pole. Thus, this projection is useful for plotting radio waves and air navigation with a compass.

The Lambert Conformal Conic projection transforms the Earth's spheroid onto a cone parallel to the Earth's axis and secant to the earth at two latitudes referred to as the standard latitudes. The East-West scale linearity is correct at the two standard latitudes and is relatively accurate in the band between these latitudes. The projection preserves direction and shape quite well within and near the standard latitudes. Hence, the Lambert, Conformal Conic Projection is best suited to East-West measurements and is useful for air navigation. Also, all meridians are straight and intersect at the pole. This

projection is most applicable to the mid-latitude region where the cone is secant to the Earth.

The Gnomonic projection transforms the Earth's spheroid onto a plane tangent to the Earth's at the point of interest. This projection is valid over all latitudes and longitudes. It has the quality of representing all great circle arcs on the projection as straight lines. Since electromagnetic waves travel the shortest distance route (great circle arc), the Gnomonic projection is ideally suited for the presentation of direction-finding lines of bearing.

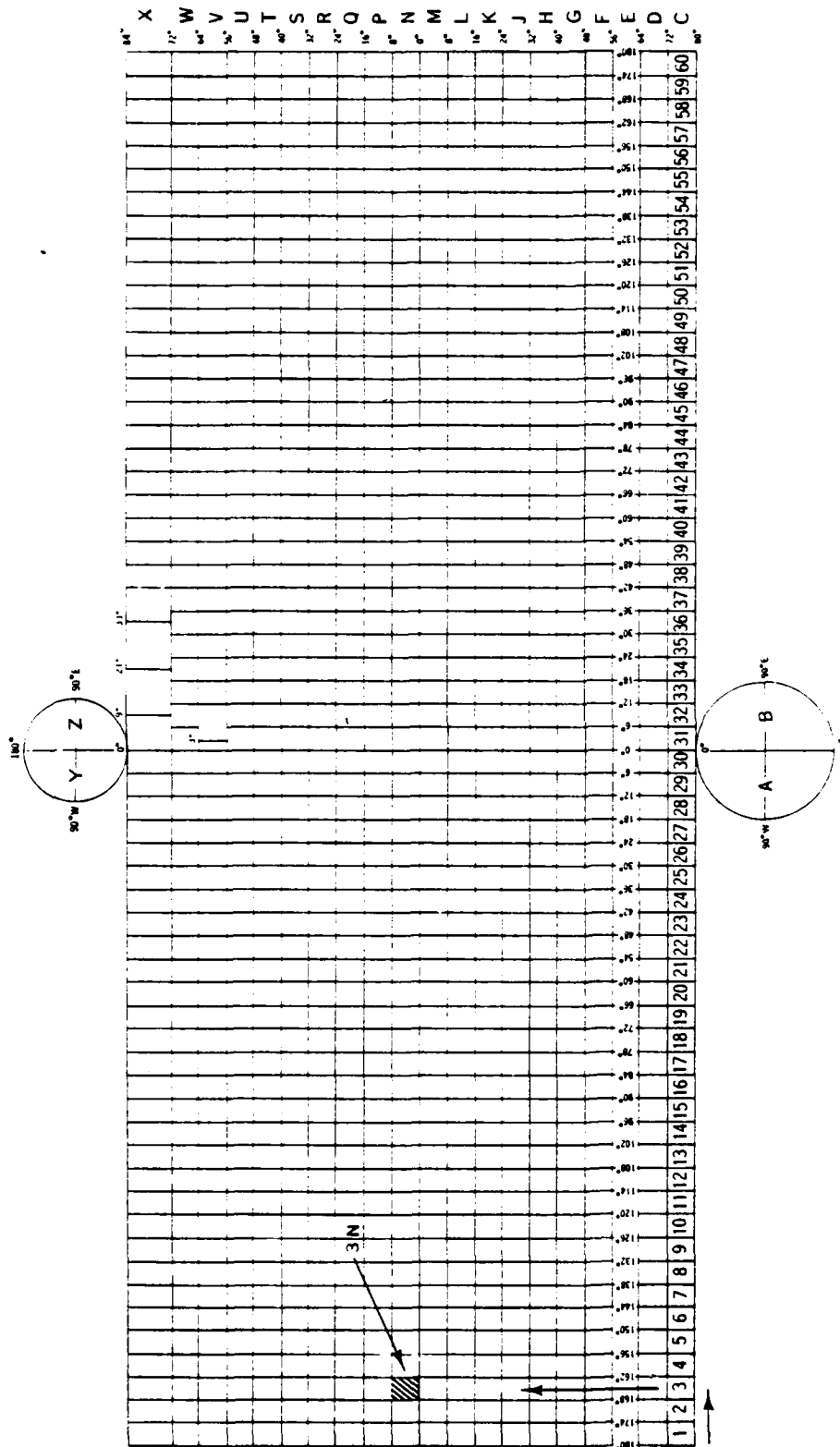
2.3. Representing Algorithms in Standard Form

To compare algorithms across systems, all algorithms analyzed must be translated into a standard format. Algorithms in the systems analyzed had been coded in such diverse languages as assembly and structured FORTRAN so that translation into a common Higher-Order Language (HOL) became essential to searching for common and diverse features. Publication ALGOL was seen as an attractive candidate because it has been used in the collected algorithms of the ACM for algorithm description. However, audiences outside the Applied Mathematics, Numerical Analysis, and Computer Sciences communities are generally unfamiliar with ALGOL; and compilers for ALGOL 60 or ALGOL 68 are not readily available in this country.

Pascal, an ALGOL-like language, has been chosen for the primary representation language for the algorithms because it has many of the properties of ALGOL (structure, strong typing, etc.), it has become familiar to a wide audience, and high quality compilers are available on many computers including the Digital Equipment Corp. VAX and many microcomputers with CPM operating systems. The last point is important because the VAX will be used by both USAICS and JPL, and the microcomputers are similar to the word processors and personal computers at JPL and USAICS. Among the features of Pascal that contribute to its clarity are the command structures, such as "if-then-else" and "case", and the user-defined data types. However, separate compilations of procedures to support hierarchical descriptions of algorithms are an implementation-dependent extension rather than a basic feature of the language. Because of this and other problems Pascal provides at best an interim solution to the algorithm description problem.

Ada offers a long-term solution. The Ada language avoids many of the shortcomings of Pascal and has many additional features. A stronger reason for using Ada is that the Army is likely to require all new systems initiated after 1984 to be programmed in it. While no complete compiler for Ada is currently available, there is an interpreter on the project VAX computer, although it is very slow. A compiler for an incomplete implementation available on Z80-based microcomputers with CPM operating systems is also available. Although this compiler is not completely satisfactory because of the lack of user-defined types, it is still useful for some simple examples and for comparison with Pascal.

THE U.S. ARMY MILITARY GRID REFERENCE SYSTEM



THE U.S. ARMY MILITARY GRID REFERENCE SYSTEM

TM 5-241-1

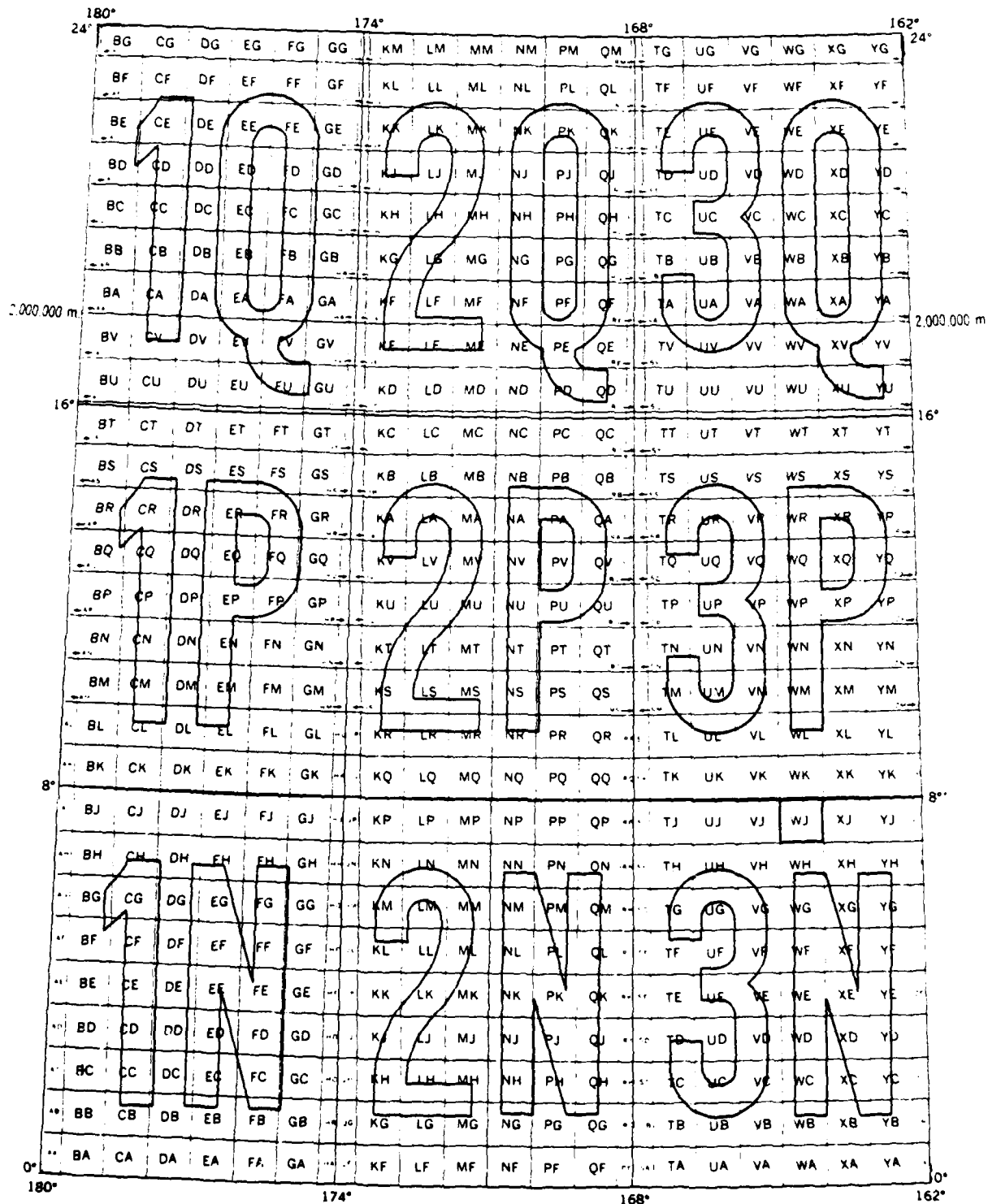


Figure 2-3: Basic Plan of the 100,000-meter Square
 Identifications of the U.S. Army Military
 Grid Reference System, between 84° N. and 80° S

3. Characterizing and Cataloging the Algorithms

The set of properties stored for algorithms in the database should characterize an algorithm for military application purposes without requiring that the algorithm itself be retrieved and examined. The algorithm property selection is influenced by the following ways the database will be used. The user may wish a general summary of what algorithms are in the database. A more likely use is to look for algorithms that perform a specified function, such as position location. In another dimension, the user may ask "What algorithms do we have in MAGIIC?". Cataloging properties should be independent for efficiency of description. Two of the properties chosen, performance and robustness, are not totally independent. Requirements performance is interpreted here as, "does the algorithm do what it says it does?". Lack of robustness is interpreted here as failure for special value or failure because of small errors in input data. Range checking of input variables is an important contributor to robustness. This is particularly important if the algorithm is to be used for more than one system where the calling programs can not be expected to protect it.

The Algorithm Level Help File; (Figure 3-1) taken from the Acquisition and Database Entry Prompting Tool (ADEPT) User's Guide, is examined and interpreted as a means for describing the present classification scheme. This information is provided for each system in which an algorithm is implemented. This set of parameters is likely to change as more experience is gained with its use.

Examples of PSA reports are shown in Appendix 7.1.

Fig. 3-1: Algorithm Level Help File

NAME is a name description of the algorithms function.

SYNONYM is the algorithms abbreviation (the same as its VAX file element name).

SOURCE:AUTHOR is the document from which the algorithm was taken and author, if known.

PROCESSING is what JPL has done with the algorithm, e.g. Pascal program tested.

MATH:FIELD is what mathematical field the algorithm is based on, e.g. least squares.

ROBUSTNESS is a measure of sensitivity to values of variables input to the algorithm, e.g. a transformation algorithm may produce an incorrect result with inputs of $\pm 180^\circ$ longitude.

TREE:LEVEL is a rough indication of location of the algorithm in the hierarchy of algorithms in a system.

REQUIREMENTS/PERFORMANCE Since the requirements documents for particular algorithms are generally not available, requirements must be derived from design documents or comments in code. This item describes how well the algorithm meets these requirements.

REFERENCE is a pointer to the VAX file containing the representation of the algorithm in standard form: Pascal and in some cases Ada. (These are given in Appendix 7.3).

4. Algorithms in MAGIIC

The MAGIIC System Lambert Constant Generation algorithm was analyzed and modeled in Pascal. This algorithm is required for analyzing and modeling the Geographic to Lambert/Polar Grid and Lambert/Polar Grid to Geographic conversions.

The MAGIIC system coordinate conversions were studied, and two inter-related algorithms were analyzed and modeled in Pascal. These were the Polar Grid to UPS and Northing and Easting to UTM conversions. Several inconsistencies have been noted (perhaps only because of the technical writing). Modeling these interrelated algorithms in Pascal led to the decision to use a non-standard Pascal feature - the VAX Pascal external procedure capability. This was considered necessary to best maintain structural integrity, accuracy and clarity in the algorithm representation.

4.1. MAGIIC Lambert Constant Generation Algorithm

The MAGIIC Lambert Constant Generation Algorithm, described in document CG108100A, dated 23 October 1978, paragraph 3.2.119, page 210, has insufficient input parameters and a lack of detail on setting the hemisphere flags to implement the algorithm in Pascal without making several assumptions. If we assume the availability of the underlying spheroid parameters, the Lambert Constant Generation Algorithm performs satisfactorily. The hemisphere flag issue remains unsettled: are they north-south hemispheres, east-west hemispheres, or both? - all three selections make sense in different contexts.

This algorithm has been modeled in Pascal on the VAX computer for uniformity of presentation and for comparison and analysis. All assumptions are included in comments in this Pascal representation (see Appendix 7.3).

4.2. An Interrelated Set of MAGIIC Coordinate Conversion Algorithms

There are a set of four interrelated coordinate conversion algorithms for MAGIIC that warrant a consolidated discussion because although they are pair-wise reciprocal and criss-cross "call" each other, their input and output parameters are specified inconsistently. These algorithms viewed as reciprocal pairs are:

1. Polar Grid to UPS (paragraph 3.2.118) reciprocal
2. UPS to Polar Grid (paragraph 3.2.117) reciprocal
3. Northing and Easting to UTM (paragraph 3.2.86) reciprocal
4. UTM to Northing and Easting (paragraph 3.2.85) reciprocal

They can also be viewed as "criss-cross" calling algorithms as follows:

1. Polar Grid to UPS (paragraph 3.2.118) each calls the other
2. Northing and Easting to UTM (paragraph 3.2.86) each calls the other
3. UPS to Polar Grid (paragraph 3.2.117) each calls the other
4. UTM to Northing and Easting (paragraph 3.2.85) each calls the other

Analysis of these algorithms and their interrelationships has revealed the following inconsistencies:

1. the Polar Grid to UPS algorithm should have an output list consistent with the UPS to Polar Grid algorithms input list since they are reciprocal functions. The lists are not consistent;
2. the Northing and Easting to UTM algorithm and the UTM to Northing and Easting algorithm, similarly, have inconsistent output and input lists;
3. in both of the above cases the same problems are true when the algorithms are used in the "criss-cross call" sense.

These points will be discussed in greater detail below in the independent analysis of the four algorithms.

4.3. MAGIIC Polar Grid to Universal Polar Sterographic Algorithm

The MAGIIC Polar Grid to Universal Polar Sterographic (UPS) Algorithm, described in document CG108100A, dated 23 October 1978, paragraph 3.2.118, page 209, fails to produce the correct output data.

This algorithm first determines if the UPS or UTM grid reference system is applicable based on the input grid zone designation. The algorithm that handles the UTM conversion is discussed as part of the Northing and Easting to Universal Transverse Mercator conversion algorithm (paragraph 3.2.86 of the previously referenced document).

In an attempt to render this algorithm amenable to analysis and modeling in Pascal several assumptions have been made (based on our interpretation of the composite of various inferences from paragraph 3.2.86, .87, .117, .118). These assumptions are:

1. The input/output lists consist of decimal and integer numbers, and alphabetic letters. There are no wholesale conversions of the input/output lists from and to ASCII representation,
2. The input/output lists are exchanged with the Northing and Easting to UTM algorithm for processing, when the input grid zone letter is from C through X (not in either polar region). Otherwise, this algorithm performs the processing (for the polar regions).

The following discussion only treats the UPS coverage area where a conversion from PS to UPS should be made. This conversion is handled within the Polar Grid to Universal Polar Sterographic conversion algorithm (paragraph 3.2.118). This algorithm is defective in producing disallowed 100,000 meter squares letter pairs.

The North Polar Area ($\geq 84^\circ$ N) and South Polar Area ($> 80^\circ$ S) are referenced using the UPS grid zone reference system with the grid zone number set to zero followed by one of the grid zone letters A, B, Y, or Z. Letter pairs represent the 100,000-meter squares which overlay the grid zone designations. It is in the lettering of these squares that this algorithm fails.

The descriptions of how to obtain the Easting letter L_E from the index I_E and the Northing letter L_N from the index I_N are merely statements that the functions should be performed without any details. Perhaps an elaborated description of the details of these functions would lead to satisfactory conversions, if these details were furnished.

4.4. MAGIIC Northing and Easting to Universal Transverse Mercator Algorithm

The MAGIIC Northing and Easting to Universal Transverse Mercator (UTM) Algorithm as described in document CG18100A, dated 23 October 1978, paragraph 3.2.86, page 156, appears to produce the correct output data on the UTM map segment available for reference. It will certainly fail in grid zones 31V, 32X, 34X, and 36X and should be further evaluated.

This algorithm first determines, whether the UTM or UPS grid reference system is applicable based on the input grid zone designation. The algorithm that handles the UPS conversion is discussed as part of the Polar Grid to Universal Polar Sterographic conversion algorithm (paragraph 3.2.118 of the previously referenced document).

In an attempt to render the algorithm amenable to analysis and to model in Pascal several assumptions have been made (based on our interpretation of the composite of various inferences from paragraph 3.2.86, .87, .117, .118). These assumptions are:

1. The input lists consist of decimal and integer numbers and alphabetic letters. There is no wholesale conversion of the input list to ASCII representation. The output lists are entirely in ASCII representation.
2. The input/output lists are exchanged with the Polar Grid to UPS algorithm for processing, when the input grid zone letter is not from C through X (for the polar regions). Otherwise, this algorithm performs the processing (for the non-polar regions).

Only the conversion to the UTM coverage area, that is, the actual conversion handled within this Northing and Easting to Universal Transverse Mercator conversion algorithm (paragraph 3.2.86), is discussed here. This

algorithm is defective in producing grid zone designation 31V, which should be truncated at 3° E, and the three non-existent grid zone designations 32X, 34X, and 36X.

5. Comparing the Algorithms Across Systems

5.1. Spheroid Models

Because the Earth is not a perfect sphere, it is modeled as a spheroid. Since this underlying spheroid model of the Earth's oblateness spans most of the coordinate conversion algorithms, the use of various spheroid models was investigated for all four systems. It was found that twelve different spheroid models were used among or in the four systems raising the possibility of inconsistencies that may hamper inter-system communication. Of these spheroid models five were used in common by all four systems. The remaining seven spheroid models were not available in all systems, as illustrated in Table 1. Some of these partially-shared spheroid models may be equivalent, but this cannot be determined until the code is available for all four systems.

5.2. Grid Zone Generation

The Grid Zone Generation algorithms were analyzed for the MAGIIC, Guardrail, Trailblazer, and BETA systems. The Guardrail Grid Zone algorithm text description is consistent with the Trailblazer text description and computer code. The MAGIIC text description differs from the Guardrail and Trailblazer descriptions and would tend to produce less efficient runtime code, but would economize memory. All three of these systems' algorithms would produce the same flawed results, except BETA which fails badly at the upper latitudes. These three versions of the Grid Zone Generation algorithm have been modeled in Pascal.

The BETA Grid Zone Generation algorithm handles details of grid zone generation more completely. The grid zone number calculations handle input longitude beyond -180° and at 180° and beyond whereas the three other systems would fail. The grid zone letter calculations handle the special conditions of grid zone truncation for grid description 31V and the non-existence of grid designations 32X, 34X, and 36X.

Pascal implementations of these algorithms are given in Appendix 7.3. The underlying assumptions are given in the comments included in the code.

5.3. MAGIIC Grid Zone Generation Algorithm

The MAGIIC Grid Zone Generation Algorithm as described in document CG108100A, dated 23 October 1978, paragraph 3.2.90, page 167 has been analyzed and found to handle the following five areas incorrectly:

1. the upper limit of longitude (180° E),
2. the latitudes $\geq 80^{\circ}$ N (considerably below the upper limit, 84° N), where it provides erroneous data,
3. the truncated grid zone 31V,
4. the non-existent grid zones 32X, 34X, and 36X,
5. the regions beyond the stated longitude and latitude limits, where it fails catastrophically.

In general, perhaps due to technical writing, there are many errors of omission and/or commission where the criteria for certain algorithm parts are left unstated; for example, a text states, "If the latitude is 84° north or greater, or 80° south or greater, the grid zone number (sic) shall be set to Y or Z or to A or B, respectively. The grid zone number shall be set to zero."

5.4. Guardrail Grid Zone Generation Algorithm

The Guardrail Grid Zone Generation Algorithm as described in document ESL-TM928, dated 15 September 1979, paragraph 16.6.2.1, page 16-192 fails in five areas:

1. at longitudes equal to and beyond 180° E and beyond 180° W.
2. at latitudes equal to and beyond 80° N and beyond 80° S,
3. at the truncated grid zone 31V,
4. at the non-existent grid zones 32X, 34X, and 36X,

5. beyond the stated longitude and latitude limits, where it fails catastrophically.

5.5. Trailblazer Grid Zone Generation Algorithm

The Trailblazer Grid Zone Generation Algorithm is described in the well-commented ROLM assembly language listings. This algorithm, extracted from the code for the GP2UM subprogram dated 20 February 1981, fails in five areas:

1. at longitudes equal to and beyond 180° E and beyond 180° N,
2. at latitudes equal to and beyond 80° N and beyond 80° S,
3. at the truncated grid zone 3IV,
4. at the non-existent grid zones 32X, 34X, and 36X,
5. beyond the stated longitude and latitude limits.

Because the hierarchical program structure is not yet available for Trailblazer, it is possible that some or all of these problems are handled adequately in higher levels of the program structure.

5.6. BETA Grid Zone Generation Algorithm

The BETA Grid Zone Generation Algorithm, described in document SS22-43, Appendix IV, page II-474 for the ADSONU subprogram, and page II-45D for the ADSCCM subprogram, was in Structured FORTRAN with in-line coding. The "INCLUDE" subprogram ZDBPRO was missing so some "reasonable" assumptions were made about it.

This algorithm performs as specified and effectively handles the following:

1. Grid zone wrap-around (longitudes $>180^{\circ}$ W or $\geq 180^{\circ}$ E),
2. North and South Polar Regions (latitudes $\geq 84^{\circ}$ N or $> 80^{\circ}$ S),

3. Truncating grid zone 31V,

4. The non-existent grid zones 32X, 34X, and 36X.

Table 5-1: Inconsistent Spheroid Usage

(X = spheroid model used in the system)

<u>Spheroid Model</u>	<u>System</u>				<u>Technical Manual</u>
	BETA	MAGIIC	GR	TB	TM241-1
Clark 1866	X	X	X	X	X
International	X	X	X	X	X
Clark 1880	X	X	X	X	X
Everest	X	X	X	X	X
Bessel	X	X	X	X	X
Australian	X	X	X		X
Walbeck				X	
Fisher	X			X	
Krasovsky			X	X	
World Geodetic	X		X		
Airy	X				
Malayan	X				
Reference	1	2	3	4	

-
1. DD2642, dtd 20 Feb 81, pg 263
 2. CG1808100A, dtd 23 Oct 78, pg 168
 3. ESL-TM929, dtd 15 Sep 79, pg 15-158
 4. TM32-5811-022-10-0

6. Discussion and Conclusions

6.1. Documentation of Algorithms

Only a design document has been available for the MAGIIC system; and the code has not been available, as shown in Figure 6-1. Therefore, some of the apparent deficiencies in the algorithms may be due to poor technical writing and may not exist in the code itself. For the Guardrail system, tapes containing code have been available, but the printouts obtained to date have been largely unreadable so that suppositions made from the documentation could not be confirmed from the code. In the case of Trailblazer, code is available, but the structured overview expected from documentation is not available.

6.2. Similarity of Functions Across Systems

The functions performed by the geographic transformation algorithms are found to be basically the same across the four systems examined, although the functions are implemented in slightly different ways.

6.3. Incompleteness of Algorithms for Global Applications

The MAGIIC, Guardrail, and Trailblazer transformation algorithms do not account for all the vagaries of the military grid system. Only the BETA algorithms account for all regions and boundaries. The former systems may ensure that "bad" arguments are never passed to these algorithms, so that no anomalies would occur in overall system performance. However, to develop a library of algorithms shared by many systems requires that algorithms internally protect themselves from "bad" input data.

6.4. Robustness of Algorithms

All systems but BETA fail to check for limits of latitude and longitude. This may arise from a common tendency to focus attention on certain areas of the world, e.g. Western Europe. This tendency is especially inappropriate when developing software that may well outlive any given political or geographical constraints.

6.5. Selection/Consolidation of Algorithms

The BETA grid zone generation algorithm is superior to those in the other three systems. Selection of a spheroid model for the library is not possible on the basis of the presently available data and may eventually require developing algorithms based on our experience with many different systems.

Fig. 6-1: Documentation of Algorithms

<u>System</u>	<u>Documentation</u>	<u>Code Available</u>	<u>Comments</u>
MAGIIC	Yes (Barely usable)	No	Bad technical writing: Errors of omission and bad-quality copy. Por- tions of some pages un- readable. TEREC task similar to Guardrail.
Guardrail	Yes Good	No Glimpses of code appear to be structured FORTRAN; most sections are missing	Multiple Tasks: Program structure "implied" by document's structure. Flowcharts with verbal descrip- tion. At least one significant technique covered by math desc- ription only, entirely included in one box at the flowchart level.
Trailblazer	No Not separately published, but basic documenta- tion included at the beginning of each code segment	Yes assembly	Well commented code. Many similarities to Guardrail.
BETA	Yes Good	Yes Comprehensive Structured FORTRAN	Program structure explicitely included in documents. Some "key" charts not read- able due to photo reductions. Code is in-line commented from Program Design Language (PDL), but we don't have the PDL

7. Appendices

7.1. Database Entries and Products (PSL/PSA)

The three attached PSA Reports were found to be useful to our analysis. The first report is essentially the PSA Report representing our PSL input data descriptions. The second is the PSA Data Activity Interaction Matrix. It shows the interrelationships between the algorithms and their input (R) and output (D) data items. The third is part of the PSA Structure Chart and is in indented hierarchy chart for the algorithms in our PSL database.

7.2. Algorithm Hierarchy Charts

Structure Report

[illegible]

SEMANTIC-MAP-TEST

[illegible]

Data Activity Interaction Matrix

```

34 MG_UTM2NE ----- /
33 MG_UPS2FG ----- /
32 MG_STEROHGHT ----- /
31 MG_RMSTARGLOC ----- /
30 MG_RMSINVTL ----- /
29 MG_RM3FLMITZN ----- /
28 MG_RMSCoordCNV ----- /
27 MG_RAD2DEG ----- /
26 MG_PROXSRCH ----- /
-----+-----+-----+
1 MG_Angle_in_Degrees/Mins/Secs | D | |
2 MG_Angle_in_Scaled_Pi_Radians | R | |
3 MG_P75T2 ----- | |
4 MG_Spheroid ----- | | D R
5 MG_P17ITS ----- | |
-----+-----+-----+
6 MG_Latitude/Longitude ----- | |
7 MG_Northings/Eastings ----- | | D F
8 MG_Grid_Zone_No./Letter ----- | | D R
9 MG_Ks_Map_Cursor_Coordinate -- | |
10 MG_UTM_Northings/Eastings_Set -- | |
-----+-----+-----+
11 MG_Grid_Zone_Letter ----- | | R
12 MG_N/E_Letters ----- | | R
13 MG_N/E_Numbers ----- | | R
14 MG_100K_Meter_Square_ID ----- | | R
-----+-----+-----+

```

	25 HG-FRIFIDEI	/
24 HG-FRECFTIE	/	/
23 MG-FG2UFS	/	/
22 HG-RE2UTH	/	/
21 MG-RE2NAFCUR	/	/
20 HG-HE2GF	/	/
19 MG-NAVCONV	/	/
18 HG-NAPIT7N	/	/
17 MG-NAFCUR2NE	/	/
16 MG-LORANCV	/	/
15 HG-LAMBERTCG	/	/
14 MG-L/FG2GF	/	/
13 MG-IIRAGLOC	/	/
12 MG-IINUVL	/	/
11 MG-IIFLMITZ	/	/
10 MG-IICORDCNV	/	/
9 MG-GRDZONECNV	/	/
8 MG-GP2NE	/	/
7 MG-GF2L/FG	/	/
6 MG-GENSPHERE	/	/
5 HG-FRANSRCH	/	/
4 MG-BEG2RAD	/	/
3 MG-BUILDSTCKH	/	/
2 MG-BUILDOP	/	/
1 MG-BEACDRF	/	/
1 MG-Angle in Degrees/Mins/Secs	R	/
2 MG-Angle in Scaled Pi-Radians	D	/
3 MG-F75T2	R R R	R R R
4 MG-Spheroid	D R	D R R R
5 MG-F171TC	R	R
6 HG-Latitude/Longitude	R K K R	D R
7 HG-Northing/Easting	D D	D R R R
8 HG-Grid-Zone-No./Letter	D D	R R R R
9 HG-Har-Cursor-Coordinate	D R D	R R R
10 MG-UTM-Northing/Easting Set	D	D
11 HG-Grid-Zone Letter		D
12 MG-M/E Letters		D
13 MG-M/F Numbers		D
14 MG-100k Meter Scale ID		D

```

1 MG_Algorithms
2   MG_BUILDCD
3     MG_IITARGLOC
4       MG_GENSPHERE
2   MG_BUILDSKTCH
3     MG_IITARGLOC
4       MG_GENSPHERE
3     MG_IIFLMITZN
3     MG_MAPCUR2NE
4       MG_GP2NE
5         MG_GENSPHERE
5         MG_GP2L/PG
5         MG_GRDZONECNV
4       MG_GRDZONECNV
4       MG_NE2GP
5         MG_GP2NE
6         MG_GENSPHERE
6         MG_GP2L/PG
6         MG_GRDZONECNV
2   MG_FRAMSRCH
3     MG_NE2GP
4       MG_GP2NE
5         MG_GENSPHERE
5         MG_GP2L/PG
5         MG_GRDZONECNV
3     MG_UTM2NE
4       MG_GENSPHERE
4       MG_NE2GP
5         MG_GP2NE
6         MG_GENSPHERE
6         MG_GP2L/PG
6         MG_GRDZONECNV
4       MG_NE2UTM
5         MG_PG2UPS
6         MG_NE2UTM
PSA168:Loop detected (see level 4) - Structure truncated.
4       MG_PG2UPS
5         MG_NE2UTM
6         MG_PG2UPS
PSA168:Loop detected (see level 4) - Structure truncated.
4       MG_UPS2PG
3     MG_IIINVTL
4       MG_IITARGLOC
5         MG_GENSPHERE
4       MG_GENSPHERE
3     MG_RMSINVTL
2   MG_IICCOORDCNV
3     MG_GP2NE
4       MG_GENSPHERE
4       MG_GP2L/PG
4       MG_GRDZONECNV
3     MG_NE2GP
4       MG_GP2NE
5         MG_GENSPHERE
5         MG_GP2L/PG
5         MG_GRDZONECNV
3     MG_UTM2NE
4       MG_GENSPHERE
4       MG_NE2GP

```

```

1 MG_Algorithms
2   MG_BUILDCD
3     MG_IITARGLOC
4     MG_GENSPHERE
2   MG_BUILDSKTC
3     MG_IITARGLOC
4     MG_GENSPHERE
3     MG_IIFLMITZN
3     MG_MAPCUR2NE
4     MG_GP2NE
5       MG_GENSPHERE
5       MG_GP2L/PG
5       MG_GRDZONECNV
4     MG_GRDZONECNV
4     MG_NE2GP
5       MG_GP2NE
6       MG_GENSPHERE
6       MG_GP2L/PG
6       MG_GRDZONECNV
2   MG_FRAMSRCH
3     MG_NE2GP
4     MG_GP2NE
5       MG_GENSPHERE
5       MG_GP2L/PG
5       MG_GRDZONECNV
3     MG_UTM2NE
4     MG_GENSPHERE
4     MG_NE2GP
5       MG_GP2NE
6       MG_GENSPHERE
6       MG_GP2L/PG
6       MG_GRDZONECNV
4     MG_NE2UTM
5       MG_PG2UPS
6       MG_NE2UTM
PSA168:Loop detected (see level 4) - Structure truncated.
4     MG_PG2UPS
5       MG_NE2UTM
6       MG_PG2UPS
PSA168:Loop detected (see level 4) - Structure truncated.
4     MG_UPS2PG
3     MG_IIINVTL
4     MG_IITARGLOC
5       MG_GENSPHERE
4     MG_GENSPHERE
3     MG_RMSINVTL
2   MG_IICOORDCNV
3     MG_GP2NE
4     MG_GENSPHERE
4     MG_GP2L/PG
4     MG_GRDZONECNV
3     MG_NE2GP
4     MG_GP2NE
5       MG_GENSPHERE
5       MG_GP2L/PG
5       MG_GRDZONECNV
3     MG_UTM2NE
4     MG_GENSPHERE
4     MG_NE2GP

```

```

5      MG_GP2NE
6      MG_GENSPHERE
6      MG_GP2L/PG
6      MG_GRDZONECNV
4      MG_NE2UTM
5      MG_PG2UPS
6      MG_NE2UTM
PSA168:Loop detected (see level 4) - Structure truncated.
4      MG_PG2UPS
5      MG_NE2UTM
6      MG_PG2UPS
PSA168:Loop detected (see level 4) - Structure truncated.
4      MG_UPS2PG
3      MG_NE2UTM
4      MG_PG2UPS
5      MG_NE2UTM
PSA168:Loop detected (see level 3) - Structure truncated.
2      MG_LORANCNV
3      MG_GENSPHERE
2      MG_MAPITZN
3      MG_LAMBERTCG
2      MG_NAVCORN
3      MG_IITARGLOC
4      MG_GENSPHERE
3      MG_IIFLHITZN
3      MG_MAP_JRCNE
4      MG_GP2NE
5      MG_GENSPHERE
5      MG_GP2L/PG
5      MG_GRDZONECNV
4      MG_GRDZONECNV
4      MG_NE2GP
5      MG_GP2NE
6      MG_GENSPHERE
6      MG_GP2L/PG
6      MG_GRDZONECNV
3      MG_GP2NE
4      MG_GENSPHERE
4      MG_GP2L/PG
4      MG_GRDZONECNV
3      MG_NE2GP
4      MG_GP2NE
5      MG_GENSPHERE
5      MG_GP2L/PG
5      MG_GRDZONECNV
3      MG_UTM2NE
4      MG_GENSPHERE
4      MG_NE2GP
5      MG_GP2NE
6      MG_GENSPHERE
6      MG_GP2L/PG
6      MG_GRDZONECNV
4      MG_NE2UTM
5      MG_PG2UPS
6      MG_NE2UTM
PSA168:Loop detected (see level 4) - Structure truncated.
4      MG_PG2UPS
5      MG_NE2UTM
6      MG_PG2UPS

```

```

PSA168:Loop detected (see level 4) - Structure truncated.
  4      MG_UPS2PG
  3      MG_NE2UTM
  4      MG_PG2UPS
  5      MG_NE2UTM
PSA168:Loop detected (see level 3) - Structure truncated.
  2      MG_PRECPTFLE
  3      MG_RMSTARGLOC
  2      MG_PRIPTDET
  3      MG_IITARGLOC
  4      MG_GENSPHERE
  3      MG_IIFLMITZN
  3      MG_MAPCUR2NE
  4      MG_GP2NE
  5      MG_GENSPHERE
  5      MG_GP2L/PG
  5      MG_GRDZONECNV
  4      MG_GRDZONECNV
  4      MG_NE2GP
  5      MG_GP2NE
  6      MG_GENSPHERE
  6      MG_GP2L/PG
  6      MG_GRDZONECNV
  3      MG_GP2NE
  4      MG_GENSPHERE
  4      MG_GP2L/PG
  4      MG_GRDZONECNV
  3      MG_NE2GP
  4      MG_GP2NE
  5      MG_GENSPHERE
  5      MG_GP2L/PG
  5      MG_GRDZONECNV
  3      MG_UTH2NE
  4      MG_GENSPHERE
  4      MG_NE2GP
  5      MG_GP2NE
  6      MG_GENSPHERE
  6      MG_GP2L/PG
  6      MG_GRDZONECNV
  4      MG_NE2UTM
  5      MG_PG2UPS
  6      MG_NE2UTM
PSA168:Loop detected (see level 4) - Structure truncated.
  4      MG_PG2UPS
  5      MG_NE2UTM
  6      MG_PG2UPS
PSA168:Loop detected (see level 4) - Structure truncated.
  4      MG_UPS2PG
  3      MG_NE2UTM
  4      MG_PG2UPS
  5      MG_NE2UTM
PSA168:Loop detected (see level 3) - Structure truncated.
  3      MG_IIINVTL
  4      MG_IITARGLOC
  5      MG_GENSPHERE
  4      MG_GENSPHERE
  3      MG_RMSINVTL
  3      MG_RMSTARGLOC
  3      MG_NE2MAPCUR

```

```

4      MG_GP2NE
5      MG_GENSPHERE
5      MG_GP2L/PG
5      MG_GRDZONECNV
4      MG_GP2L/PG
4      MG_NE2GP
5      MG_GP2NE
6      MG_GENSPHERE
6      MG_GP2L/PG
6      MG_GRDZONECNV
2      MG_PROXSRCH
3      MG_MAPCUR2NE
4      MG_GP2NE
5      MG_GENSPHERE
5      MG_GP2L/PG
5      MG_GRDZONECNV
4      MG_GRDZONECNV
4      MG_NE2GP
5      MG_GP2NE
6      MG_GENSPHERE
6      MG_GP2L/PG
6      MG_GRDZONECNV
3      MG_GP2NE
4      MG_GENSPHERE
4      MG_GP2L/PG
4      MG_GRDZONECNV
3      MG_UTM2NE
4      MG_GENSPHERE
4      MG_NE2GP
5      MG_GP2NE
6      MG_GENSPHERE
6      MG_GP2L/PG
6      MG_GRDZONECNV
4      MG_NE2UTM
5      MG_PG2UPS
6      MG_NE2UTM
PSA168:Loop detected (see level 4) - Structure truncated.
4      MG_PG2UPS
5      MG_NE2UTM
6      MG_PG2UPS
PSA168:Loop detected (see level 4) - Structure truncated.
4      MG_UPS2PG
2      MG_STEROHGHT
3      MG_IITARGLOC
4      MG_GENSPHERE
3      MG_IIFLMITZN
3      MG_IIINVTL
4      MG_IITARGLOC
5      MG_GENSPHERE
4      MG_GENSPHERE

```

4 MG_DEG2RAD

PROCESS

DESCRIPTION:

This algorithm converts angle in degrees/minutes/seconds into its equivalent angle in scaled PI radians.

SOURCES: CG106100A/Part_1

SECURITY: U

RESP PD: JWG

ATTRIBUTE:

VALUE:

abbreviation	Degrees_2_Binary_6Binary
type_of_source	document
date_acquired	'07/01/82'
processing_done	none
mathematical_field	trigonometry
tree_level	leaf
requirements_performance	TBD

DESCRIPTION:

This algorithm determines the appropriate spheroid number corresponding to the input latitude/longitude by scanning a specially constructed map database which contains the relationship between longitude bands/latitude strips and spheroid numbers.

KEYWORDS: algorithm

SOURCES: 3.2.91

CG108100A/Part-I

SECURITY: U

RESP PD: JWG

ATTRIBUTE:

VALUE:

abbreviation	Spheroid_Generation
type_of_source	document
date_acquired	'07/01/82'
processing_done	none
mathematical_field	cartography
tree_level	leaf
requirements_performance	TBD

DESCRIPTION:

This algorithm converts an input geographic coordinate latitude/longitude pair to the equivalent Lambert/polar grid northings/easting coordinate set.

KEYWORDS: algorithm

SOURCES: 3.2.115

CG108100A/Part-I

SECURITY: U

RESP FD: JWG

ATTRIBUTE:

VALUE:

abbreviation	Lat/Long-2-Lambert/Polar_Grid
type_of_source	document
date_acquired	'07/01/82'
processing_done	none
mathematical_field	cartography
tree_level	leaf
requirements_performance	TBD

8 MG-GP2NE

PROCESS

DESCRIPTION:

This algorithm converts and inputs latitude, longitude pair into its equivalent northing and easting coordinate set.

KEYWORDS: algorithm

SOURCES: 3.2.83

CG108100A/Part-I

SECURITY: U

RESP PD: JWG

ATTRIBUTE:

VALUE:

abbreviation	Geographic_2_Northings/Easting
type_of_source	document
date_acquired	'07/01/82'
processing_done	none
mathematical_field	cartography
tree_level	leaf
requirements_performance	TBD

DESCRIPTION:

This algorithm converts input longitude and latitude to UTM/UPS Grid Zone number and letter. It fails at the upper (closed) limits of both longitude and latitude.

KEYWORDS: algorithm

SOURCES: 3.2.90

CG108100A/Part I

SECURITY: U

RESF PD: JWG

ATTRIBUTE:

VALUE:

abbreviation	Grid_Zone_Generation
type_of_source	document
date_acquired	'07/01/82'
processing_done	analyzed/Pascalized
mathematical_field	cartography
robustness	3
tree_level	leaf
requirements_performance	generally_satisfactory
references	'CJWG1QRUGZHG/MGGZDG'

DESCRIPTION:

This algorithm converts Lambert/polar grid input coordinates into equivalent geographic coordinate latitude/longitude.

KEYWORDS: algorithm

SOURCES: 3.2.116

CG108100A/Part_I

SECURITY: U

RESP PD: JWG

ATTRIBUTE:

VALUE:

abbreviation	Lambert/Polar_Grid_2_Lat/Long
type_of_source	document
date_acquired	'07/01/82'
processing_done	none
mathematical_field	cartography
tree_level	leaf
requirements_performance	TBD

15 M0-LAMBERTCG

PROCESS

DESCRIPTION:

This algorithm calculates the Lambert constants required for use with the Lambert/Polar Grid to UPS and the UPS to Lambert/Polar Grid conversion algorithms.

KEYWORDS: algorithm

SOURCES: 3.2.119

CG106100A/Part-I

SECURITY: U

RESP PD: JWG

ATTRIBUTE:

VALUE:

abbreviation	LambertConstantGeneration
type_of_source	document
date_acquired	'07/01/82'
processing_done	analysed/Fascicized
mathematical_field	cartographics
robustness	6
tree_level	leaf
requirements_performance	satisfactory
references	'1JWG103PROC

DESCRIPTION:

This algorithm converts an xy map cursor position to its equivalent northings/easting or latitude/longitude or grid zone number/letter and spheroid output coordinate.

KEYWORDS: algorithm

SOURCES: 3.2.89

CG108100A/Part_I

SECURITY: U

RESP PD: JWG

ATTRIBUTE:

VALUE:

abbreviation	Map_Cursor12_Northings_Easting
type_of_source	document
date_acquired	'07/01/82'
processing_done	none
mathematical_field	cartography
tree_level	middle
requirements_performance	TBD

DESCRIPTION:

This algorithm converts and inputs northings/eastings set into a latitude/longitude pair.

KEYWORDS: algorithm

SOURCES: 3.2.84

CG108100A/Part_I

SECURITY: U

RESP PD: JWG

ATTRIBUTE:

VALUE:

abbreviation	Northings/Eastings/Offset/Long
type_of_source	document
date_acquired	'07/01/82'
processing_done	none
mathematical_field	cartographic
tree_level	leaf
requirements_performance	TBD

01 MG_NE2MAPCUR

PROCESS

DESCRIPTION:

This algorithm converts either northings/eastings or latitude/longitude or grid zone number/letter pairs into equivalent x-y map cursor position pairs for display.

KEYWORDS: algorithm

SOURCES: 3.2.88

CG108100A/Part-I

SECURITY: U

RESP PD: JWG

ATTRIBUTE:

VALUE:

abbreviation	Northings/Eastings-2-Line Cursor
type_of_source	document
date_acquired	'07/01/82'
processing_done	none
mathematical_field	cartography
tree_level	middle
requirements_performance	TBD

22 MG_NE2UTM

PROCESS

DESCRIPTION:

This algorithm converts northings and eastings to a composite UTM pair.

KEYWORDS: algorithm

SOURCES: 3.2.86

CG108100A/Part_I

SECURITY: U

RESP FD: JWG

ATTRIBUTE:

VALUE:

abbreviation	Northings/Eastings-22UTM
type_of_source	document
date_acquired	107/01/82
processing_done	none
mathematical_field	cartography
tree_level	leaf
requirements_performance	TBD

3 MG_PG2UPS

PROCESS

DESCRIPTION:

This algorithm converts Polar Grid northing/easting coordinates into equivalent Universal Polar Stereographic or Universal Transverse Mercator (utilizing the NE2UTM algorithm).

KEYWORDS: algorithm

SOURCES: 3.2.118

CG108100A/Part_I

SECURITY: U

RESP PD: JWG

ATTRIBUTE:

VALUE:

abbreviation	Polar_Grid_2_UPS
type_of_source	document
date_acquired	'07/01/82'
processing_done	analyzed/Pascalized
mathematical_field	cartography
robustness	3
tree_level	leaf
requirements_performance	TBD
references	'CGJWG3PG2UPS'

DESCRIPTION:

This algorithm converts an angle in scaled Pi radians into an equivalent angle in degrees/minutes/seconds.

SOURCES: CG108100A/Part_I

SECURITY: U

RESP PD: JWG

ATTRIBUTE:

VALUE:

abbreviation	Radians_2_Degrees
type_of_source	document
date_acquired	'07/01/82'
processing_done	none
mathematical_field	trigonometry
tree_level	leaf
requirements_performance	TBD

DESCRIPTION:

This algorithm converts Universal Polar Stereographic into equivalent polar grid coordinates.

KEYWORDS: algorithm

SOURCES: 3.2.117

CG108100A/Part_I

SECURITY: U

RESP FD: JWG

ATTRIBUTE:

VALUE:

abbreviation	UPS_2_Polar_Grid
type_of_source	document
date_acquired	'07/01/82'
processing_done	none
mathematical_field	cartography
tree_level	leaf
requirements_performance	TBD

DESCRIPTION:

This algorithm converts a UTM coordinate set into the equivalent composite northing and easting pair.

KEYWORDS: algorithm

SOURCES: 3.2.65

CG108100A/Part-I

SECURITY: U

RESP PD: JWG

ATTRIBUTE:

VALUE:

abbreviation	UTM_2_Northings/Eastings
type_of_source	document
date_acquired	'07/01/82'
processing_done	none
mathematical_field	cartographs
tree_level	leaf
requirements_performance	TED

7.3. Algorithms in Standard Form

```

100 PROGRAM DriverLambertConstantGeneration (INPUT,OUTPUT);
200 {
300 { DriverLambertConstantGeneration provides an environment for
400 { DriverLambertConstantGeneration PROCEDURE
500 { testing the LambertConstantGeneration procedure
600 {
700 { Interfaces with the LambertConstantGeneration procedure are:
800 { GLOBAL VARIABLES - Eccentricity
900 { SemimajorAxis
1000 { INPUT PARAMETERS - LambdaL
1100 { LambdaR
1200 { PhiU
1300 { PhiL
1400 { Phi1
1500 { Phi2
1600 { OUTPUT PARAMETERS - Kappa
1700 { Iota
1800 { ProjectionConeRadius
1900 { LambdaC
2000 { SquaredEccentricity
2100 { Hemisphere
2200 {
2300 { declarations
2400 {
2500 TYPE
2600 PiRadians = REAL;
2700 ZeroToOne = REAL;
2800 Flag = (northern:southern);
2900 {
3000 VAR
3100 Eccentricity : ZeroToOne;
3200 SemimajorAxis : REAL;
3300
3400 LambdaL : PiRadians;
3500 LambdaR : PiRadians;
3600 PhiU : PiRadians;
3700 PhiL : PiRadians;
3800 Phi1 : PiRadians;
3900 Phi2 : PiRadians;
4000 {
4100 Kappa : REAL;
4200 Iota : REAL;
4300 ProjectionConeRadius : REAL;
4400 LambdaC : PiRadians;
4500 SquaredEccentricity : ZeroToOne;
4600 Hemisphere : Flag;
4700 {
4800 {
4900 PROCEDURE LambertConstantGeneration
5000 ( {GLOBAL} Eccentricity : ZeroToOne;
5100 {GLOBAL} SemimajorAxis : REAL;
5200 {IN} LambdaL : PiRadians;
5300 {IN} LambdaR : PiRadians;
5400 {IN} PhiU : PiRadians;
5500 {IN} PhiL : PiRadians;
5600 {IN} Phi1 : PiRadians;
5700 {IN} Phi2 : PiRadians;
5800 {OUT} VAR Kappa : REAL;
5900 {OUT} VAR Iota : REAL;
6000 {OUT} VAR ProjectionConeRadius : REAL;
6100 {OUT} VAR LambdaC : PiRadians;
6200 {OUT} VAR SquaredEccentricity : ZeroToOne;
6300 {OUT} VAR Hemisphere : Flag ) EXTERNAL;

```

```

6400 {
6500 { Procedure LambertConstantGeneration models the Lambert Constant }
6600 { Generation algorithm described in paragraph 3.2.119 of document }
6700 { CG108100a dated 23 October 1978. }
6800 {
6900 { J.W.Gillis 4-22-82 }
7000 {
7100 { This procedure ASSUMES that certain data are available as }
7200 { required by the algorithm but not described as inputs in the }
7300 { reference document. These data are: Eccentricity }
7400 { SemimajorAxis }
7500 {
7600 {
7700 { This procedure DOESNOT perform data validation checks that are }
7800 { not specified in the algorithm description. This is to allow the }
7900 { algorithm features to be presented more clearly. }
8000 {
8100 { executables }
8200 {
8300 BEGIN
8400 {
8500 { establish GLOBAL variables }
8600 {
8700 { Eccentricity := 0.5; }
8800 { SemimajorAxis := 10.0; }
8900 {
9000 { establish INPUT PARAMETERS }
9100 {
9200 { LambdaL := 0.25; }
9300 { LambdaR := 0.0; }
9400 { PhiU := 0.125; }
9500 { PhiL := 0.0; }
9600 { Phi1 := 0.0675; }
9700 { Phi2 := 0.375; }
9800 {
9900 { echo INPUT PARAMETERS }
10000 {
10100 { WRITELN; }
10200 { WRITELN (' setup Eccentricity is ',Eccentricity); }
10300 { WRITELN (' setup SemimajorAxis is ',SemimajorAxis); }
10400 { WRITELN (' setup LambdaL is ',LambdaL); }
10500 { WRITELN (' setup LambdaR is ',LambdaR); }
10600 { WRITELN (' setup PhiU is ',PhiU); }
10700 { WRITELN (' setup PhiL is ',PhiL); }
10800 { WRITELN (' setup Phi1 is ',Phi1); }
10900 { WRITELN (' setup Phi2 is ',Phi2); }
11000 {
11100 { LambertConstantGeneration ( (GLOBAL) Eccentricity, }
11200 { (GLOBAL) SemimajorAxis, }
11300 { (OUT) LambdaL, }
11400 { (OUT) LambdaR, }
11500 { (OUT) PhiU, }
11600 { (OUT) PhiL, }
11700 { (OUT) Phi1, }
11800 { (OUT) Phi2, }
11900 { (IN) Kappa, }
12000 { (IN) Iota, }
12100 { (IN) ProjectionConeRadius, }
12200 { (IN) LambdaC, }
12300 { (IN) SquaredEccentricity, }
12400 { (IN) Hemisphere ); }
12500 {
12600 { list output parameters from last }

```

```

12700      (
12800          WRITELN;
12900          WRITELN (' Kappa is ' ,Kappa);
13000          WRITELN (' Iota is ' ,Iota);
13100          WRITELN (' ProjectionConeRadius is ' ,ProjectionConeRadius);
13200          WRITELN (' LambdaC is ' ,LambdaC);
13300          WRITELN (' SquaredEccentricity is ' ,SquaredEccentricity);
13400          WRITELN (' Hemisphere is ' ,Hemisphere)
13500      (
13600  END. ( DriverLambertConstantGeneration PROCEDURE

```

```

100  MODULE LesProc (INPUT,OUTPUT);
200  (
300  (
400  (
500  TYPE
600      zeroToOne          = REAL;
700      PiRadians          = REAL;
800      Flag                = (northern,southern);
900  (
1000 (
1100 (
1200 PROCEDURE LambertConstantGeneration
1300     ( (GLOBAL) Eccentricity      : ZeroToOne;
1400       (GLOBAL) SemimajorAxis     : REAL;
1500       (IN)      LambdaL          : PiRadians;
1600       (IN)      LambdaR          : PiRadians;
1700       (IN)      PhiU             : PiRadians;
1800       (IN)      PhiL             : PiRadians;
1900       (IN)      Phi1             : PiRadians;
2000       (IN)      Phi2             : PiRadians;
2100       (OUT) VAR Kappa            : REAL;
2200       (OUT) VAR Iota             : REAL;
2300       (OUT) VAR ProjectionConeRadius : REAL;
2400       (OUT) VAR LambdaC          : PiRadians;
2500       (OUT) VAR SquaredEccentricity : ZeroToOne;
2600       (OUT) VAR Hemisphere       : Flag );
2700 (
2800 ( Generation algorithm described in paragraph 3.2.119 of document
2900 ( CG108100a dated 23 October 1978.
3000 (
3100 ( J.W.Gillis 4-27-82
3200 (
3300 ( This procedure ASSUMES that certain data are available as
3400 ( required by the algorithm but not described as inputs in the
3500 ( reference document. These data are: Eccentricity
3600 (                               SemimajorAxis
3700 (
3800 ( This procedure DOESNOT perform data validation checks that are
3900 ( not specified in the algorithm description. This is to allow the
4000 ( algorithm features to be presented more clearly.
4100 (
4200 (
4300 (
4400 CONST
4500     PiOver2          = 1.57079;
4600 (
4700 VAR
4800     Phi              : ARRAY [1..3] OF PiRadians;
4900     PhiPrime         : ARRAY [1..3] OF PiRadians;
5000     Zeta             : ARRAY [1..3] OF PiRadians;
5100     Curvature        : ARRAY [1..2] OF REAL;
5200     Index            : INTEGER;
5300 BEGIN
5400 (
5500     Phi[1] := Phi1;
5600     Phi[2] := Phi2;
5700     Phi[3] := (PhiU+PhiL)/2.0;
5800 (
5900     LambdaC := (LambdaL + LambdaR)/2.0;
6000 (
6100     SquaredEccentricity := SQR(Eccentricity);
6200 (
6300     FOR Index :=1 TO 3 DO

```

```

6400      PhiPrime[Index] := ARCTAN((1.0-SquaredEccentricity)*
6500      (SIN(ABS(Phi[Index]))/
6600      COS(ABS(Phi[Index]))));
6700  (
6800      FOR Index := 1 TO 2 DO
6900          Curvature[Index] := SemiMajorAxis/
7000          (SQRT(1.0-SquaredEccentricity*
7100          SQR(SIN(ABS(Phi[Index])))));
7200      FOR Index := 1 TO 3 DO
7300          Zeta[Index] := PiOver2-PhiPrime[Index];
7400  (
7500      Iota := (LN(COS(ABS(Phi[1])))-LN(COS(ABS(Phi[2])))+
7600      LN(Curvature[1])-LN(Curvature[2]))/
7700      (LN(SIN(Zeta[1]/2.0)/COS(Zeta[1]/2.0))-
7800      LN(SIN(Zeta[2]/2.0)/COS(Zeta[2]/2.0)));
7900  (
8000      Kappa := (Curvature[1]*COS(ABS(Phi[1]))/
8100      (Iota*((SIN(Zeta[1]/2.0)/COS(Zeta[1]/2.0))**Iota));
8200  (
8300      ProjectionConeRadius := Kappa*((SIN(Zeta[3]/2.0)/
8400      COS(Zeta[3]/2.0))**Iota);
8500  (
8600      { NO ALGORITHM is available for the Hemisphere Flag set }
8700  (
8800      { list intermediate values }
8900  (
9000      WRITELN;
9100      WRITELN (' computed Phi[3] is ',Phi[3]);
9200      FOR Index := 1 TO 3 DO
9300          WRITELN (' computed PhiPrime[Index] is ',PhiPrime[Index]);
9400      FOR Index := 1 TO 2 DO
9500          WRITELN (' computed Curvature[Index] is ',Curvature[Index]);
9600      FOR Index := 1 TO 3 DO
9700          WRITELN (' computed Zeta[Index] is ',Zeta[Index]);
9800  (
9900      END; { LambertConstantGeneration PROCEDURE }
10000  (
10100      END. { LogProc MODULE }

```

```

100  (
200  PROGRAM Drvit (INPUT,OUTPUT);
300  (
400  (
500  TYPE
600      Meters          =REAL;
700      Decameters      =REAL;
800      Letter          ='A'..'Z';
900      Spheres         =INTEGER;
1000 (
1100 VAR    PGNorthingsCoord    :Meters;
1200         PGEastingsCoord    :Meters;
1300         PGZoneLetter       :Letter;
1400         PGZoneNumber       :INTEGER;
1500         SpheroidNumber     :Spheres;
1600         UPSZoneLetter      :Letter;
1700         UPEastingsLetter   :Letter;
1800         UPSNorthingsLetter :Letter;
1900         UPEastingsNumber    :Decameters;
2000         UPSNorthingsNumber :Decameters;
2100 (
2200 (
2300 PROCEDURE PolarToUPS
2400     (IN )    PGNorthingsCoord    :Meters;
2500     (IN )    PGEastingsCoord    :Meters;
2600     (IN )    PGZoneLetter       :Letter;
2700     (IN )    PGZoneNumber       :INTEGER;
2800     (IN )    SpheroidNumber     :Spheres;
2900     (OUT) VAR UPSZoneLetter      :Letter;
3000     (OUT) VAR UPEastingsLetter   :Letter;
3100     (OUT) VAR UPSNorthingsLetter :Letter;
3200     (OUT) VAR UPEastingsNumber    :Decameters;
3300     (OUT) VAR UPSNorthingsNumber :Decameters);EXTERN;
3400 (
3500     BEGIN
3600 (
3700     WRITELN;
3800     WRITELN (' ENTER PG NORTHING COORD ');
3900     READLN (PGNORTHINGCOORD);
4000     WRITELN;
4100     WRITELN (' ENTER PG EASTING COORD ');
4200     READLN (PGEASTINGCOORD);
4300     WRITELN;
4400     WRITELN (' ENTER PG ZONE NUMBER ');
4500     READLN (PGZONENUMBER);
4600     WRITELN;
4700     WRITELN (' ENTER PG ZONE LETTER ');
4800     READLN (PGZONELETTER);
4900     WRITELN;
5000     WRITELN (' ENTER SPHEROID NUMBER ');READLN (SPHEROIDNUMBER);
5100 (
5200 (
5300     PolarToUPS (PGNorthingsCoord,
5400                 PGEastingsCoord,
5500                 PGZoneLetter,
5600                 PGZoneNumber,
5700                 SpheroidNumber,
5800                 UPSZoneLetter,
5900                 UPEastingsLetter,
6000                 UPSNorthingsLetter,
6100                 UPEastingsNumber,
6200                 UPSNorthingsNumber);
6300 (

```

6400 END. (of PROGRAM Drvlt)

```

100  (
200  MODULE PG2UPS (INPUT,OUTPUT);
300  (
400  (
500  TYPE
600      Meters          =REAL;
700      Decameters      =REAL;
800      Letter          ='A'..'Z';
900      Spheres         =INTEGER;
1000 (
1100 (
1200 (
1300 PROCEDURE PolarToUPS
1400     (IN )      PGNorthingsCoord      :Meters;
1500     (IN )      PGEastingsCoord       :Meters;
1600     (IN )      PGZoneLetter           :Letter;
1700     (IN )      PGZoneNumber           :INTEGER;
1800     (IN )      SpheroidNumber         :Spheres;
1900     (OUT) VAR   UPSZoneLetter         :Letter;
2000     (OUT) VAR   UPSEastingsLetter     :Letter;
2100     (OUT) VAR   UPSNorthingsLetter    :Letter;
2200     (OUT) VAR   UPSEastingsNumber     :Decameters;
2300     (OUT) VAR   UPSNorthingsNumber    :Decameters);
2400 (
2500 (
2600 ( PROCEDURE PolarToUPS models the Polar Grid to Universal Polar
2700 ( Stereographic conversion algorithm described in Paragraph
2800 ( 3.2.116 of CG108100A dated 23 October 1978.
2900 (
3000 ( PROCEDURE PolarToUPS CALLs: PROCEDURE ConvertToUPS
3100 (                               PROCEDURE ConvertToUTM
3200 (
3300 ( Programmed by J.W.Gillis      5-5-82
3400 (
3500 ( This procedure ASSUMES that certain data are available as
3600 ( required by the algorithm but not adequately described in the
3700 ( algorithm description.
3800 (
3900 ( This procedure DOESNOT perform data validity checks that are
4000 ( not specified in the algorithm description. This is to allow
4100 ( the algorithm features to be presented more clearly.
4200 (
4300 ( PROCEDURE PolarToUPS accepts northing and easting coordinates,
4400 ( tests whether the UPS or UTM grid system is a suitable target
4500 ( system. If the UPS grid system is appropriate, the the
4600 ( conversion is performed by this procedure. If the UTM grid
4700 ( system is appropriate, then this procedure CALLs the
4800 ( PolarToUTM procedure (algorithm 3.2.86) to perform the
4900 ( conversion.
5000 (
5100 (
5200 (
5300 TYPE
5400     Meters          = REAL;
5500     Decameters      = REAL;
5600     Letter          = 'A'..'Z';
5700     Spheres         = INTEGER;
5800 (
5900 VAR
6000     UTMZoneLetter   :Letter;
6100     UTMEastingsLetter :Letter;
6200     UTMNorthingsLetter :Letter;
6300     UTMEastingsNumber :Decameters;

```

```

6400         UTMNorthingsNumber    :Decameters;
6500     (
6600 PROCEDURE ConvertToUPS
6700         (IN )      FGNorthingsCoord    :Meters;
6800         (IN )      FGEastingsCoord     :Meters;
6900         (IN )      FGZoneLetter        :Letter;
7000         (IN )      FGZoneNumber       :INTEGER;
7100         (OUT) VAR UPSZoneLetter        :Letter;
7200         (OUT) VAR UPSEastingsLetter     :Letter;
7300         (OUT) VAR UPSNorthingsLetter    :Letter;
7400         (OUT) VAR UPSEastingsNumber     :Decameters;
7500         (OUT) VAR UPSNorthingsNumber    :Decameters);EXTERN;
7600     (
7700 PROCEDURE ConvertToUTM
7800         (IN )      FGNorthingsCoord    :Meters;
7900         (IN )      FGEastingsCoord     :Meters;
8000         (IN )      FGZoneLetter        :Letter;
8100         (IN )      FGZoneNumber       :INTEGER;
8200         (IN )      SpheroidNumber      :Spheres;
8300         (OUT) VAR UTMZoneLetter        :Letter;
8400         (OUT) VAR UTMEastingsLetter     :Letter;
8500         (OUT) VAR UTMNorthingsLetter    :Letter;
8600         (OUT) VAR UTMEastingsNumber     :Decameters;
8700         (OUT) VAR UTMNorthingsNumber    :Decameters);EXTERN;
8800     (
8900     (
9000         (Select the appropriate grid reference system
9100         (
9200         BEGIN
9300         WRITELN (' FG2UPS ALL HOOKED UP ');
9400         IF FGZoneNumber = 0
9500             THEN ConvertToUPS ((OUT) FGNorthingsCoord,
9600                                (OUT) FGEastingsCoord,
9700                                (OUT) FGZoneLetter,
9800                                (OUT) FGZoneNumber,
9900                                (IN ) UPSZoneLetter,
10000                               (IN ) UPSEastingsLetter,
10100                               (IN ) UPSNorthingsLetter,
10200                               (IN ) UPSEastingsNumber,
10300                               (IN ) UPSNorthingsNumber)
10400             ELSE ConvertToUTM ((OUT) FGNorthingsCoord,
10500                                (OUT) FGEastingsCoord,
10600                                (OUT) FGZoneLetter,
10700                                (OUT) FGZoneNumber,
10800                                (OUT) SpheroidNumber,
10900                                (IN ) UTMZoneLetter,
11000                                (IN ) UTMEastingsLetter,
11100                                (IN ) UTMNorthingsLetter,
11200                                (IN ) UTMEastingsNumber,
11300                                (IN ) UTMNorthingsNumber);
11400         (
11500         (
11600         END; (of PROCEDURE PolarToUPS)
11700         (
11800         (
11900         END. (of MODULE FG2UPS)

```

```

100  (
200  MODULE CNV2UPS (INPUT,OUTPUT);
300  (
400  (
500  TYPE
600      Meters          =REAL;
700      Decameters      =REAL;
800      Letter          ='A'..'Z';
900  (
1000 PROCEDURE ConvertToUPS
1100      (IN )      PGNorthingCoord      :Meters;
1200      (IN )      PGEastingCoord       :Meters;
1300      (IN )      PGZoneLetter         :Letter;
1400      (IN )      PGZoneNumber          :INTEGER;
1500      (OUT) VAR  UPSZoneLetter         :Letter;
1600      (OUT) VAR  UPSEastingLetter      :Letter;
1700      (OUT) VAR  UPSNorthingLetter    :Letter;
1800      (OUT) VAR  UPSEastingNumber      :Decameters;
1900      (OUT) VAR  UPSNorthingNumber     :Decameters);
2000 (
2100 (
2200 ( PROCEDURE ConvertToUPS performs the Polar Grid to Universal
2300 ( Polar Stereographic conversion algorithm described in
2400 ( Paragraph 3.2.118 of CG108100A dated 23 October 1978.
2500 (
2600 ( PROCEDURE ConvertToUPS is CALLED by: PROCEDURE PolarToUPS
2700 (                                     PROCEDURE PolarToUTM
2800 (
2900 ( Programmed by J.W.Gillis          5-6-82
3000 (
3100 ( This procedure ASSUMES that certain data are available as
3200 ( required by the algorithm but not adequately described in the
3300 ( algorithm description.
3400 (
3500 ( This procedure DOESNOT perform data validity checks that are
3600 ( not specified in the algorithm description. This is to allow
3700 ( the algorithm features to be presented more clearly.
3800 (
3900 ( PROCEDURE ConvertToUPS accepts polar grid northings and eastings
4000 ( coordinates and converts them to UPS coordinates.
4100 (
4200 (
4300 TYPE
4400     Meters          = REAL;
4500     Decameters      = REAL;
4600     Letter          = 'A'..'Z';
4700 (
4800 VAR
4900     Index            : INTEGER;
5000     GridLetter      : ARRAY [1..26] OF Letter ;
5100 (
5200 FUNCTION aMODb ((IN) a,b:REAL):INTEGER;
5300 (
5400     VAR Ainteger      : INTEGER;
5500     Binteger          : INTEGER;
5600 (
5700 BEGIN
5800     Ainteger:=TRUNC(a);
5900     Binteger:=TRUNC(b);
6000     aMODb:=Ainteger MOD Binteger
6100 END; (or FUNCTION MODab)
6200 (
6300 (

```

```

6400      (
6500      BEGIN
6600      (
6700      (
6800      {Initialize the GridLetter array.
6900      (
7000      FOR Index := 1 TO 26 DO
7100          GridLetter[Index]:=CHR(ORD('A')+Index-1);
7200      (
7300      (
7400      (
7500      {Calculate the UPSEastingNumber & UPSEastingLetter Index
7600      (
7700      IF PGZoneLetter > 'M'
7800          THEN BEGIN
7900          UPSEastingNumber:=aMODb((PGEastingCoord-200000),
8000              100000)/10;
8100          Index:=aMODb((PGEastingCoord-200000)/100000,20)
8200      END; {of IF}
8300      (
8400      IF PGZoneLetter < 'N'
8500          THEN BEGIN
8600          UPSEastingNumber:=aMODb(PGEastingCoord,
8700              100000)/10;
8800          Index:=aMODb(PGEastingCoord/100000,16)
8900      END; {of IF}
9000      (
9100      WRITELN;
9200      WRITELN ('UPSEASTINGNUMBER IS ',UPSEASTINGNUMBER);
9300      WRITELN ('AND ITS LETTER INDEX IS ',INDEX);
9400      (
9500      {Calculate the UPSEastingLetter from its index.
9600      (
9700      IF Index > 16
9800          THEN Index:=Index+2;
9900      IF (Index > 2) AND (Index < 17)
10000      THEN Index:=Index+1;
10100      UPSEastingLetter:=GridLetter[Index];
10200      (
10300      WRITELN;
10400      WRITELN ('UPSEASTINGLETTER IS ',UPSEASTINGLETTER);
10500      WRITELN ('AND ITS CORRECTED INDEX IS ',INDEX);
10600      (
10700      (
10800      {Calculate the UPSNorthingsNumber & NorthingsLetter Index
10900      (
11000      IF PGZoneLetter > 'M'
11100          THEN BEGIN
11200          UPSNorthingsNumber:=aMODb((PGNorthingsCoord-1300000),
11300              100000)/10;
11400          Index:=aMODb(((PGNorthingsCoord-1300000)/100000),
11500              24)
11600      END; {of IF}
11700      (
11800      IF PGZoneLetter < 'N'
11900          THEN BEGIN
12000          UPSNorthingsNumber:=aMODb((PGNorthingsCoord-800000),
12100              100000)/10;
12200          Index:=aMODb(((PGNorthingsCoord-800000)/100000),
12300              24)
12400      END; {of IF}
12500      (
12600      (

```

```

12700  WRITELN;
12800  WRITELN ('UPSNORTHINGNUMBER IS ',UPSNORTHINGNUMBER);
12900  WRITELN ('AND ITS LETTER INDEX IS ',INDEX);
13000      {Calculate the UPSNorthingLetter from its Index      }
13100      {
13200      IF Index > 16
13300          THEN Index:=Index+2;
13400      IF Index < 17
13500          THEN INDEX:=Index+1;
13600      UPSNorthingLetter:=GridLetter[Index];
13700      {
13800  WRITELN;
13900  WRITELN ('UPSNORTHINGLETTER IS ',UPSNORTHINGLETTER);
14000  WRITELN ('AND ITS CORRECTED INDEX IS ',INDEX);
14100      {
14200      {
14300      {
14400      {
14500  END; {of PROCEDURE ConvertToUPS}
14600      {
14700  END. {of MODULE CNV2UPS}

```

TEMPORARY

STUB PROCEDURE

```

100  (
200  MODULE CNV2UTM (INPUT,OUTPUT);
300  (
400  (
500  TYPE
600      Meters          =REAL;
700      Decameters      =REAL;
800      Letter          =SET OF CHAR;
900      Spheres         =INTEGER;
1000 (
1100 (
1200 PROCEDURE ConvertToUTM
1300     (IN )      PGNorthingCoord    :Meters;
1400     (IN )      PGEastingCoord     :Meters;
1500     (IN )      PGZoneLetter        :Letter;
1600     (IN )      PGZoneNumber        :INTEGER;
1700     (IN )      SPheroidNumber      :Spheres;
1800     (OUT) VAR   UTMZoneLetter       :Letter;
1900     (OUT) VAR   UTMEastingLetter    :Letter;
2000     (OUT) VAR   UTMNorthingLetter   :Letter;
2100     (OUT) VAR   UTMEastingNumber    :Decameters;
2200     (OUT) VAR   UTMNorthingNumber   :Decameters;
2300 (
2400 (
2500 ( PROCEDURE ConvertToUTM performs the Polar Grid to Universal
2600 ( Transverse Mercator conversion algorithm described in
2700 ( Paragraph 3.2.86 of CG108100A dated 23 October 1978.
2800 (
2900 ( PROCEDURE ConvertToUTM is CALLED by: PROCEDURE PolarToUTM
3000 (                                     PROCEDURE PolarToUPS
3100 (
3200 ( Programmed by J.w.sillis          5-6-82
3300 (
3400 ( This procedure ASSUMES that certain data are available as
3500 ( required by the algorithm but not adequately described in the
3600 ( algorithm description.
3700 (
3800 ( This procedure DOESNOT perform data validity checks that are
3900 ( not specified in the algorithm description. This is to allow
4000 ( the algorithm features to be presentes more clearly.
4100 (
4200 ( PROCEDURE ConvertToUTM accepts polar grid northings and eastings
4250 ( coordinates and converts them to UTM coordinates.
4300 (
4400 (
4500 (
4600 (
4700 (
4800 (
4900 (
5000 (
5100 TYPE
5200     Meters          = REAL;
5300     Decameters      = REAL;
5400     Letter          = SET OF CHAR;
5500     Spheres         = INTEGER;
5600 (
5700 (
5800 BEGIN
5900 ( IF
6000 (THEN)
6002 WRITELN (' CNV2UTM HOOKED UP OK')
6100 (
6200 (
6300 END; (of PROCEDURE ConvertToUTM)
6400 (
6500 END. (of MODULE Cnv2UTM)

```

```

100  C)
200  PROGRAM ConvertToUTM (INPUT,OUTPUT);
300  C)
400  TYPE
500      ASCIIArray      = ARRAY [1..4] OF CHAR;
600      Letter          = CHAR;
700  C)
800  VAR
900      UTMCoordinate    : RECORD
1000          UTMZoneNumber : ARRAY[1..2] OF INTEGER;
1100          UTMZoneLetter : Letter;
1200          UTMEastingsLetter : Letter;
1300          UTMNorthingsLetter : Letter;
1400          UTMEastingsNumber : ARRAY[1..4] OF CHAR;
1500          UTMNorthingsNumber : ARRAY[1..4] OF CHAR;
1600      END; {UTMC }
1700  C)
1800      PGNorthingsCoord : INTEGER; {IN}
1900      PGEastingsCoord  : INTEGER; {IN}
2000      PGZoneLetter     : Letter; {IN}
2100      PGZoneNumber     : INTEGER; {IN}
2200      SpheroidNumber    : INTEGER; {IN}
2300      I, EletterOffset, Tel, Te, Tnl: INTEGER; {Local}
2400      Number           : INTEGER;
2500  C)
2600  C)
2700  C) PROCEDURE ConvertToUTM performs the Polar Grid to Universal  >
2800  C) Transverse Mercator conversion algorithm described in  >
2900  C) paragraph 3.2.86 of CG108100A dated 23 October 1978.  >
3000  C)
3100  C) PROCEDURE ConvertToUTM is CALLED by: PROCEDURE PolarToUTM  >
3200  C)                                     PROCEDURE PolarToUPS  >
3300  C)
3400  C)
3500  C) This procedure ASSUMES that certain data are available as  >
3600  C) required by the algorithm but not adequately described in the >
3700  C) algorithm description.  >
3800  C)
3900  C) This procedure DOESNOT perform data validity checks that are >
4000  C) not specified in the algorithm description. This is to allow >
4100  C) the algorithm features to be presented more clearly.  >
4200  C)
4300  C) PROCEDURE ConvertToUTM accepts Polar grid northings and eastings >
4400  C) coordinates and converts them to UTM coordinates.  >
4500  C)
4600  C)
4700      PROCEDURE ConvertToASCII( Number: INTEGER {in};
4800          VAR AlphaNum:ASCIIArray );
4900  C)
5000      C) Convert an integer number to its ASCII representation in base 10 >
5100      C) This procedure not detailed in source >
5200  C)
5300      VAR      I: INTEGER;
5400              Adj: INTEGER;
5500  C)
5600      BEGIN
5700          FOR I := 4 DOWNTO 1 DO
5800              BEGIN
5900                  Adj := (Number MOD 10) + ORD('0') ;
6000                  AlphaNum[I] := CHR(Adj) ;
6100                  Number := Number DIV 10
6200              END
6300  C)

```

```

6400     END; (ConvertToASCII);
6500     ();
6600     ();
6700     BEGIN (ConvertToUTM);
6800     ();
6900     ( read input );
7000     ();
7100     WRITELN;
7200     WRITELN ('PG Northing Coord (integer) ');
7300     READLN (PGNorthingCoord);
7400     WRITELN ('PG Easting Coord (integer) ');
7500     READLN (PGEastingCoord);
7600     WRITELN ('PG Zone Letter (Letter) ');
7700     READLN (PGZoneLetter);
7800     WRITELN ('PG Zone Number (integer) ');
7900     READLN (PGZoneNumber);
8000     WRITELN ('Spheroid Number (integer) ');
8100     READLN (SpheroidNumber);
8200     ();
8300     ( end of input );
8400     ();
8500     WITH UTMCoordinate DO
8600     BEGIN
8700     ( Find Easting Letter (A2) );
8800     I := PGZoneNumber MOD 3;
8900     ( Easting Letter Offset Factor );
9000     CASE I OF
9100     0: ELetterOffset := 24;
9200     1: ELetterOffset := 6;
9300     2: ELetterOffset := 15;
9400     END;
9500     ();
9600     Tel := ( PGEastingCoord - 500000 ) DIV 100000 + ELetterOffset;
9700     ();
9800     ( Following letter conversion not according to documentation; );
9900     ( Procedure in documentation fails at least at Bad Hersfeld, FRG, );
10000    ( which is in 32U, and this one succeeds at least there. );
10100    ();
10200    IF Tel > 15 THEN Tel := Tel + 2;
10300    ELSE IF Tel > 10 THEN Tel := Tel - 1;
10400    ELSE Tel := Tel - 2;
10500    Tel := Tel + ORD('A') - 1;
10600    UTMEastingLetter := Chr( Tel ) (A2);
10700    ();
10800    ( Find Northing Number );
10900    IF NOT ODD( PGZoneNumber ) THEN
11000    PGNorthingCoord := PGNorthingCoord + 500000;
11100    Tn := PGNorthingCoord MOD 2000000;
11200    Tn := Tn DIV 100000 + 1;
11300    ( Make Spheroid Adjustment );
11400    CASE SpheroidNumber OF
11500    ();
11600    ( Clark 1866 );
11700    1: IF (PGZoneNumber < 31) OR (PGZoneNumber > 58) THEN Tn1 := Tn1 + 10;
11800    ELSE IF (PGZoneNumber > 51) AND (PGZoneNumber < 59) (also in source)
11900    THEN Tn1 := Tn1 - 10;
12000    ( International );
12100    2: IF (PGZoneNumber > 46) AND (PGZoneNumber < 52)
12200    THEN Tn1 := Tn1 + 10;
12300    ();
12400    ( Clark 1880 );
12500    3: Tn1 := Tn1 + 10;
12600    ();

```

```

12700      { Everest }
12800      4: IF PGZoneNumber < 46 THEN Tn1 := Tn1 + 10;
12900      {}
13000      { Bessel }
13100      5: IF (PGZoneNumber<32) AND (PGZoneLetter>'R') THEN Tn1 := Tn1 + 10;
13200      {}
13300      { Australian }
13400      6: IF ODD( PGZoneNumber ) THEN Tn1 := Tn1 + 15
13500          ELSE Tn1 := Tn1 + 5
13600      {}
13700      END; {CASE OF SpheroidNumber}
13800      {}
13900      { Calculate Final Northings Letter Index }
14000      {}
14100      IF Tn1 > 14 THEN Tn1 := Tn1 + 2
14200          ELSE IF Tn1 > 9 THEN Tn1 := Tn1 + 1;
14300      Tn1 := Tn1 + ORD('A') - 1 ;
14400      UTMNorthingsLetter := CHR( Tn1 );
14500      { Format for Output }
14600      ConvertToASCII( PGNorthingsCoord MOD 100000, UTMNorthingsNumber );
14700      ConvertToASCII( PGEastingCoord MOD 100000, UTMEastingNumber );
14800      UTMZoneNumber[1] := PGZoneNumber DIV 10;
14900      UTMZoneNumber[2] := PGZoneNumber MOD 10 ;
15000      UTMZoneLetter := PGZoneLetter ;
15100      {}
15200      { WRITE OUTPUT }
15300      {}
15400      WRITELN ;
15500      WRITELN ;
15600      WRITELN;
15700      FOR I := 1 TO 2 DO WRITELN ( 'UTM Zone Number ',I,UTMZoneNumber[I]);
15800      WRITELN ( 'UTM Zone Letter      ',UTMZoneLetter) ;
15900      WRITELN ( 'UTM Easting Letter   ',UTMEastingLetter) ;
16000      WRITELN ( 'UTM Northings Letter ',UTMNorthingsLetter) ;
16100      FOR I := 1 TO 4 DO BEGIN
16200      WRITELN ( 'UTM Easting Number  ',I,'      ',UTMEastingNumber[I]);
16300      WRITELN ( 'UTM Northings Number ',I,'      ',UTMNorthingsNumber[I]);
16400      END ;
16500      {}
16600      { END OF OUTPUT }
16700      {}
16800      END {OP WITH UTM }
16900      {}
17000      END. {ConvertToUTM}

```

```

PROCEDURE ConvertToUTM IS
--
TYPE ShortArray IS ARRAY(1..4) OF CHARACTER;
--
    PGNorthingCoord : INTEGER; --IN
    PGEastingCoord  : INTEGER; --IN
    PGZoneLetter     : CHARACTER; --IN
    PGZoneNumber     : INTEGER; --IN
    SpheroidNumber   : INTEGER; --IN
--
    UTMZoneNumber    : ARRAY(1..2) OF INTEGER;
    UTMZoneLetter    : CHARACTER;
    UTMEastingLetter : CHARACTER;
    UTMNorthingLetter : CHARACTER;
    UTMEastingNumber : ARRAY(1..4) OF CHARACTER;
    UTMNorthingNumber : ARRAY(1..4) OF CHARACTER;
--
    I, EletterOffset, Tel, Tp, tnl, ASCIIPos: INTEGER; --Local
--
PROCEDURE ConvertToASCII( Number: in INTEGER;
                          AlphaNum: out ShortArray) IS
-- Convert an integer number to its ASCII representation in base 10
-- This procedure not detailed in source
BEGIN
    FOR I in reverse 1 .. 4 LOOP
        ASCIIPos := Number MOD 10 + CHARACTER'POS('0');
        AlphaNum(i) := CHARACTER'VAL( ASCIIPos );
        Number := Number / 10;
    END LOOP;
END ConvertToASCII;
FUNCTION ODD( I: INTEGER ) return BOOLEAN is
BEGIN
    IF I = 2 * ( I/2 ) THEN return false;
    Else return true;
END IF;
END ODD;
BEGIN
-- Find Easting Letter (A2)
I := PGZoneNumber MOD 3;
-- Easting Letter Offset Factor
CASE I IS
    WHEN 0=>EletterOffset := 24;
    WHEN 1=>EletterOffset := 6;
    WHEN 2=>EletterOffset := 15;
    WHEN OTHERS => NULL;
END CASE;
--
Tel := ( PGEastingCoord - 500000 ) / 100000 + EletterOffset;
IF Tel > 14 THEN Tel := Tel + 2;
    ELSIF Tel > 9 THEN Tel := Tel + 1;
END IF;
Tel := Tel + CHARACTER'POS('a') - 1;
UTMEastingLetter := CHARACTER'VAL( Tel ); --A2
--
-- Find Northing Number
IF NOT Odd( PGZoneNumber ) THEN
    PGNorthingCoord := PGNorthingCoord + 500000;
END IF;

```

```

Tp := PGNorthingCoord MOD 2000000;
Tn1 := Tp / 100000 + 1;
-- Make Spheroid Adjustment
CASE SpheroidNumber IS
  -- Clark 1866
  WHEN 1=>IF PGGridzoneNum <31 OR PGZoneNumber > 58 THEN Tn1 := Tn1 + 10;
            ELSIF PGZoneNumber > 51 AND PGZoneNumber < 59 --typo in source
            THEN Tn1 := Tn1 - 10;
            END IF;
  -- International
  WHEN 2=>IF PGZoneNumber > 46 AND PGZoneNumber < 52 THEN Tn1 := Tn1 + 10;
            END IF;
  -- Clark 1880
  WHEN 3=>Tn1 := Tn1 + 10;
  -- Everest
  WHEN 4=>IF PGZoneNumber < 46 THEN Tn1 := Tn1 + 10; END IF;
  -- Bessel
  WHEN 5=>IF PGZoneNumber <52 AND PGZoneLetter <'R' THEN Tn1 := Tn1 + 10;
            END IF;
  -- Australian
  WHEN 6=>IF ODD( PGZoneNumber ) THEN Tn1 := Tn1 + 15;
            END IF;
  WHEN OTHERS => NULL;      -- Not in source
END CASE;
--
-- Calculate Final Northing Letter Index
If Tn1 > 14 Then Tn1 := Tn1 + 2;
  ELSIF Tn1 > 9 THEN Tn1 := Tn1 + 1;
END IF;
Tn1 := Tn1 + CHARACTER'POS('A') - 1;
UTMNorthingLetter := CHARACTER'VAL( Tn1 );
-- format for output
ConvertToASCII( PGNorthingCoord MOD 100000, UTMNorthingNumber );
ConvertToASCII( PGEastingCoord MOD 100000, UTMEastingNumber );
UTMZoneNum(1) := PGZoneNumber / 10;
UTMZoneNum(2) := PGZoneNumber MOD 10;
UTMZoneLetter := PGZoneLetter;
END ConvertToUTM;

```

```

100  {
200  PROGRAM DRVGZTB ( INPUT,OUTPUT );
300  {
400  { PROGRAM DRVGZTB provides a test driver capability for testing }
500  { GridZoneGeneration procedures for TRAILBLAZER. }
600  {
700  TYPE
800      DegreesReal      = Real;
900      ZoneRange       = 1..60;
1000     Letters         = 'A'..'Z';
1100  VAR
1200     Longitude         : DegreesReal;
1300     Latitude          : DegreesReal;
1400     GridZoneNumber    : ZoneRange;
1500     GridZoneLetter    : Letters;
1600  {
1700  PROCEDURE TBGridZoneGeneration
1800      (OUT) Longitude      : DegreesReal;
1900      (OUT) Latitude       : DegreesReal;
2000      (IN ) VAR GridZoneNumber : ZoneRange;
2100      (IN ) VAR GridZoneLetter : Letters);EXTERN;
2200  {
2300  { PROCEDURE TBGridZoneGeneration models the TRAILBLAZER conversion }
2400  { of geographic coordinates to Universal Transverse Mercator }
2500  { (UTM) coordinates grid zone designator number,letter. }
2600  {
2700  BEGIN
2800  {
2900      WRITELN ( ' ENTER Longitude' );
3000      READLN ( Longitude );
3100      WRITELN ( ' ENTER Latitude' );
3200      READLN ( Latitude );
3300      TBGridZoneGeneration ( (OUT) Longitude,Latitude,
3400                           (IN ) gridzonenumber,gridzoneletter);
3500      WRITELN;
3600      WRITELN ( ' GridZoneNumber is ',GridZoneNumber );
3700      WRITELN;
3800      WRITELN ( ' GridZoneLetter is ',GridZoneLetter );
3900  {
4000  END. { of PROGRAM DRVGZTB }

```

```

100  (
200  MODULE TBGZDG ( INPUT,OUTPUT );
300  (
400  TYPE
500      DegreesReal          = REAL;
600      ZoneRange            = 1..60;
700      Letters               = 'A'..'Z';
800  (
900  PROCEDURE TBGridZoneGeneration
1000      ( (IN ) Longitude      : DegreesReal;
1100      ( (IN ) Latitude       : DegreesReal;
1200      ( (OUT) VAR GridZoneNumber : ZoneRange;
1300      ( (OUT) VAR GridZoneLetter : Letters);
1400  (
1500  ( PROCEDURE TBGridZoneGeneration models the TRAILBLAZER conversion?
1600  ( of geographic coordinates to Universal Transverse Mercator
1700  ( (UTM) coordinates - grid zone designator number, letter.
1800  (
1900  ( Documentation used was the GP20K subprogram dtd. 20 feb 81
2000  ( from the listings provided for the TRAILBLAZER system.
2100  (
2200  ( PROCEDURE TBGridZoneGeneration is referenced by:
2300  (     PROGRAM DRUGZTR
2400  (
2500  ( PROCEDURE TBGridZoneGeneration makes no references.
2600  (
2700  ( This procedure DOES NOT perform any data validity checks
2800  ( that are not explicitly specified in the algorithm
2900  ( description. This is to allow the algorithm features to be
3000  ( represented more clearly.
3100  (
3200  TYPE
3300      Letters          = 'A'..'Z';
3400      IndexRange       = 1..24;
3500  (
3600  VAR
3700      GridZoneLtrList  : ARRAY[1..24] OF LETTERS;
3800      GridZoneIndex    : IndexRange;
3900  (
4000  BEGIN
4100  ( Initialize allowable characters array
4200  (
4300      GridZoneLtrList [1]    := 'A';
4400      GridZoneLtrList [2]    := 'B';
4500      GridZoneLtrList [3]    := 'C';
4600      GridZoneLtrList [4]    := 'D';
4700      GridZoneLtrList [5]    := 'E';
4800      GridZoneLtrList [6]    := 'F';
4900      GridZoneLtrList [7]    := 'G';
5000      GridZoneLtrList [8]    := 'H';
5100      GridZoneLtrList [9]    := 'J';
5200      GridZoneLtrList [10]   := 'K';
5300      GridZoneLtrList [11]   := 'L';
5400      GridZoneLtrList [12]   := 'M';
5500      GridZoneLtrList [13]   := 'N';
5600      GridZoneLtrList [14]   := 'P';
5700      GridZoneLtrList [15]   := 'Q';
5800      GridZoneLtrList [16]   := 'R';
5900      GridZoneLtrList [17]   := 'S';
6000      GridZoneLtrList [18]   := 'T';
6100      GridZoneLtrList [19]   := 'U';
6200      GridZoneLtrList [20]   := 'V';
6300      GridZoneLtrList [21]   := 'W';

```

```

6400      GridZoneLtrList [22]      := 'X';
6500      GridZoneLtrList [23]      := 'Y';
6600      GridZoneLtrList [24]      := 'Z';
6700  {
6800      GridZoneNumber := TRUNC (31.0+(Longitude/6.0));
6900      GridZoneIndex  := TRUNC (13.0+(Latitude/6.0));
7000      GridZoneLetter := GridZoneLtrList[GridZoneIndex];
7100  {
7200  END; { of PROCEDURE TSGridZoneGeneration }
7300  {
7400  END. { of MODULE GRGZTB }

```

```

100  {
200  PROGRAM DRUGZMG ( INPUT,OUTPUT );
300  {
400  { PROGRAM DRUGZMG Provides a test driver capability for testing }
500  { GridZoneGeneration procedures for MAGIIC. }
600  {
700  TYPE
800      Radians          = REAL;
900      ZoneRange       = 1..60;
1000     Letters         = 'A'..'Z';
1100  VAR
1200     Longitude        : Radians;
1300     Latitude         : Radians;
1400     GridZoneNumber   : ZoneRange;
1500     GridZoneLetter   : Letters;
1600  {
1700  PROCEDURE MGGridZoneGeneration
1800      (OUT) Longitude    : Radians;
1900      (OUT) Latitude    : Radians;
2000      (IN ) VAR GridZoneNumber : ZoneRange;
2100      (IN ) VAR GridZoneLetter : Letters); EXTERNAL;
2200  {
2300  { PROCEDURE MGGridZoneGeneration models the MAGIIC conversion }
2400  { of geographic coordinates to Universal Transverse Mercator }
2500  { (UTM) coordinates grid zone designator number & letter. }
2600  {
2700  BEGIN
2800  {
2900      WRITELN ( ' ENTER Longitude' );
3000      READLN ( Longitude );
3100      WRITELN ( ' ENTER Latitude' );
3200      READLN ( Latitude );
3300      MGGridZoneGeneration ( (OUT) Longitude, Latitude,
3400                           (IN ) GridZoneNumber, GridZoneLetter );
3500      WRITELN;
3600      WRITELN ( ' GridZoneNumber is ', GridZoneNumber );
3700      WRITELN;
3800      WRITELN ( ' GridZoneLetter is ', GridZoneLetter );
3900  {
4000  END. { OF PROGRAM DRUGZMG }

```

```

100  {
200  MODULE MGGZDG ( INPUT,OUTPUT );
300  {
400  TYPE
500      Radians                = REAL;
600      ZoneRange              = 1..60;
700      Letters                = 'A'..'Z';
800  {
900  PROCEDURE MGGridZoneGeneration
1000      (IN )      Longitude      : Radians;
1100      (IN )      Latitude       : Radians;
1200      (OUT) VAR  GridZoneNumber : ZoneRange;
1300      (OUT) VAR  GridZoneLetter : Letters);
1400  {
1500  { PROCEDURE MGGridZoneGeneration models the MAGIIC conversion
1600  { of geographic coordinates to Universal Transverse Mercator
1700  { (UTM) coordinates - grid zone designator number & letter.
1800  {
1900  { Documentation used was source code listings from the MAGIIC
2000  { document CG108100A dtd. 23 Oct 1978, par.3.2.90, pg.167.
2100  {
2200  { PROCEDURE MGGridZoneGeneration is referenced by:
2300  {     PROGRAM DRV6ZMG
2400  {
2500  { PROCEDURE MGGridZoneGeneration makes no references.
2600  {
2700  { This procedure DOES NOT perform any data validity checks
2800  { that are not explicitly specified in the algorithm
2900  { description. This is to allow the algorithm features to be
3000  { represented more clearly.
3100  {
3200      CONST
3300          Pi                = 3.1415926;
3400  {
3500      TYPE
3600          IndexRange        = 1..26;
3700  {
3800      VAR
3900          GridZoneIndex      : IndexRange;
4000  {
4100      BEGIN
4200  {
4300  { Calculate the grid zone number
4400  {
4500  { STATED LONGITUDE RANGE IS -180<=LONGITUDE<=180 IN DEGREES
4600  {
4700      GridZoneNumber := TRUNC(((180.0/Pi)*Longitude+180.0)/6.0)+1;
4800  {
4900  { NO compensation for wrap around of grid zone numbers
5000  {
5100  { Determine the grid zone letter
5200  {
5300  { Since no details are provided in the referenced documentation
5400  { it is ASSUMED that we know how to assign A or B for latitudes
5500  { equal to or over 84 degrees North and Y or Z for latitudes
5600  { equal to or over 80 degrees South.
5700  {
5800  { TRUNCATION to integer is ASSUMED since it is necessary at this
5900  { point in order to use the GridZoneIndex as a pointer.
6000  {
6100  { STATED LATITUDE RANGE IS 80<=LATITUDE<=84 IN DEGREES

```

```

6200  {
6300      GridZoneIndex := TRUNC (((180.0/Pi)*Latitude+90.0)/8.0);

6400  {
6500  { Compute midrange grid zone letters
6600  {
6700      IF GridZoneIndex <= 5
6800      THEN GridZoneLetter := CHR(GridZoneIndex + ORD('C'));
6900  { Here we handle the 'i' which is not used
7000  {
7100      IF ( GridZoneIndex >= 6 ) AND
7200      ( GridZoneIndex <= 10 )
7300      THEN GridZoneLetter := CHR(GridZoneIndex + ORD('C')+1);
7400  {
7500  { Here we handle the 'O' which is not used
7600  {
7700      IF ( GridZoneIndex >= 11 ) AND
7800      ( GridZoneIndex <= 19 )
7900      THEN GridZoneLetter := CHR(GridZoneIndex + ORD('C')+2);
8000  {
8100  { The rest of the GridZoneIndex are biased off by ORD ('C')
8200  {
8300      IF GridZoneIndex > 19
8400      THEN GridZoneLetter := CHR(GridZoneIndex);
8500  {
8600  { Assign Y or Z to the North Polar Zone according as Western
8700  { or Eastern Hemisphere, respectively.
8800  {
8900      IF Latitude*(180.0/Pi) >= 84.0
9000      THEN IF Longitude*(180.0/Pi) < 0.0
9100          THEN GridZoneLetter := 'Y'
9200          ELSE GridZoneLetter := 'Z';
9300  {
9400  { Assign A or B to the South Polar Zone according as Western
9500  { or Eastern Hemisphere, respectively.
9600  {
9700      IF Latitude*(180.0/Pi) <= -80.0
9800      THEN IF Longitude*(180.0/Pi) < 0.0
9900          THEN GridZoneLetter := 'A'
10000         ELSE GridZoneLetter := 'B';
10100  {
10200  { NO correction for the four irregular zones -
10300  {   32X,34X, and 36X do not exist
10400  {   31V is truncated
10500  {
10600  END; { of PROCEDURE MGGridZoneGeneration
10700  {
10800  END. { of MODULE MGGZDG

```

```

100  {
200  PROGRAM DRVGZBT ( INPUT,OUTPUT );
300  {
400  { PROGRAM DRVGZBT provides a test driver capability for testing }
500  { GridZoneGeneration Procedures for BETA. }
600  {
700  TYPE
800      Radians                = REAL;
900      ZoneRange              = 1..60;
1000     Letters                = 'A'..'Z';
1100  VAR
1200      Longitude              : Radians;
1300      Latitude               : Radians;
1400      GridZoneNumber         : ZoneRange;
1500      GridZoneLetter         : Letters;
1600      CenterMeridian         : Radians;
1700  {
1800  PROCEDURE BTGridZoneGeneration
1900      (OUT) Longitude          : Radians;
2000      (OUT) Latitude          : Radians;
2100      (IN ) VAR GridZoneNumber : ZoneRange;
2200      (IN ) VAR GridZoneLetter  : Letters;
2300      (IN ) VAR CenterMeridian : Radians);EXTERN;
2400  {
2500  { PROCEDURE BTGridZoneGeneration models the BETA conversion }
2600  { of geographic coordinates to Universal Transverse Mercator }
2700  { (UTM) coordinates grid zone designator number, letter and the }
2800  { central meridian of the rectangle. }
2900  {
3000  BEGIN
3100  {
3200      WRITELN ( ' ENTER Longitude' );
3300      READLN ( Longitude );
3400      WRITELN ( ' ENTER Latitude' );
3500      READLN ( Latitude );
3600      BTGridZoneGeneration ( (OUT) Longitude, Latitude,
3700                          (IN ) GridZoneNumber, GridZoneLetter,
3800                          CenterMeridian );
3900      WRITELN;
4000      WRITELN ( ' GridZoneNumber is ', GridZoneNumber );
4100      WRITELN;
4200      WRITELN ( ' GridZoneLetter is ', GridZoneLetter );
4300      WRITELN;
4400      WRITELN ( ' CenterMeridian is ', CenterMeridian );
4500  {
4600  END. { of PROGRAM DRVGZBT }

```

```

100  (
200  MODULE BTGZDG ( INPUT,OUTPUT ) ;
300  (
400  TYPE
500      Radians          = REAL;
600      ZoneRange        = 1..60;
700      Letters          = 'A'..'Z';
800  (
900  PROCEDURE BTGridZoneGeneration
1000      ( (IN ) Longitude      : Radians;
1100      ( (IN ) Latitude       : Radians;
1200      ( (OUT) VAR GridZoneNumber : ZoneRange;
1300      ( (OUT) VAR GridZoneLetter : Letters;
1400      ( (OUT) VAR CenterMeridian : Radians);
1500  (
1600  ( PROCEDURE BTGridZoneGeneration models the BETA conversion
1700  ( of geographic coordinates to Universal Transverse Mercator
1800  ( (UTM) coordinates - grid zone designator number, letters, and
1900  ( the central meridian.
2000  (
2100  ( Documentation used was source code listings from the BETA
2200  ( document SS22-43 dtd. 16 Oct 1981, pp.473.2-474 for the
2300  ( ADSCNU subprogram and pg.2-450 for the ADSCOM subprogram
2400  (
2500  ( PROCEDURE BTGridZoneGeneration is referenced by:
2600  (     PROGRAM DRUGZBT
2700  (
2800  ( PROCEDURE BTGridZoneGeneration makes no references.
2900  (
3000  ( This procedure DOES NOT perform any data validity checks
3100  ( that are not explicitly specified in the algorithm
3200  ( description. This is to allow the algorithm features to be
3300  ( represented more clearly.
3400  (
3500  ( Since the included 'ZDBPRO.COM' is not available to us at
3600  ( this time, we assume implicit typing in the source FORTRAN
3700  ( code.
3800  (
3900  CONST
4000      Pi          = 3.1415926;
4100  (
4200  TYPE
4300      Letters      = 'A'..'Z';
4400      IndexRange   = 1..24;
4500  (
4600  VAR
4700      GridZoneLtrList : ARRAY[1..24] OF LETTERS;
4800      GridZoneIndex    : IndexRange;
4900  (
5000  BEGIN
5100  ( Initialize allowable characters array
5200  (
5300      GridZoneLtrList [1]      := 'A';
5400      GridZoneLtrList [2]      := 'B';
5500      GridZoneLtrList [3]      := 'C';
5600      GridZoneLtrList [4]      := 'D';
5700      GridZoneLtrList [5]      := 'E';
5800      GridZoneLtrList [6]      := 'F';
5900      GridZoneLtrList [7]      := 'G';
6000      GridZoneLtrList [8]      := 'H';
6100      GridZoneLtrList [9]      := 'J';
6200      GridZoneLtrList [10]     := 'K';
6300      GridZoneLtrList [11]     := 'L';

```

```

6400      GridZoneLtrList [12]      := 'M';
6500      GridZoneLtrList [13]      := 'N';
6600      GridZoneLtrList [14]      := 'P';
6700      GridZoneLtrList [15]      := 'Q';
6800      GridZoneLtrList [16]      := 'R';
6900      GridZoneLtrList [17]      := 'S';
7000      GridZoneLtrList [18]      := 'T';
7100      GridZoneLtrList [19]      := 'U';
7200      GridZoneLtrList [20]      := 'V';
7300      GridZoneLtrList [21]      := 'W';
7400      GridZoneLtrList [22]      := 'X';
7500      GridZoneLtrList [23]      := 'Y';
7600      GridZoneLtrList [24]      := 'Z';
7700      (
7800      ( Calculate the grid zone number
7900      (
8000          GridZoneNumber := (TRUNC(((18000.0/PI)*Longitude)
8100                               +18600.0)) DIV 60;
8200      (
8300      ( Compensate for wrap around of grid zone numbers
8400      (
8500          IF GridZoneNumber > 60
8600          THEN GridZoneNumber := GridZoneNumber-60;
8700          IF GridZoneNumber < 1
8800          THEN GridZoneNumber := GridZoneNumber+60;
8900      (
9000      ( Determine the grid zone letter
9100      (
9200          GridZoneIndex := TRUNC((Latitude*(180.0/PI)+104.0)/8.0);
9300      (
9400      ( Test for and lock out North Polar Zones
9500      (
9600          IF GridZoneIndex > 22
9700          THEN GridZoneIndex := 22;
9800      (
9900      ( NOTE that no such test is needed for the South Polar Zone
10000      ( because the algorithm limit was given as <= 80 South
10100      (
10200      (
10300          GridZoneLetter := GridZoneLtrList[GridZoneIndex];
10400      (
10500      ( Correct for the four irregular zones -
10600      ( 32X, 34X, and 36X do not exist
10700      ( 31V is truncated
10800      (
10900      ( Truncate grid zone 31V
11000      (
11100          IF (GridZoneIndex = 20) AND
11200             ((GridZoneNumber = 31) AND
11300              (Longitude >= 3.0*(PI/180.0)))
11400          THEN GridZoneNumber := 32;
11500      (
11600      ( Correct for grid zones 32X, 34X, and 36X
11700      (
11800          IF (GridZoneIndex = 22) AND
11900             (GridZoneNumber = 32)
12000          THEN IF Longitude >= 9.0*(PI/180.0)
12100                 THEN GridZoneNumber := 33
12200                 ELSE GridZoneNumber := 31;
12300      (
12400
12500          IF (GridZoneIndex = 22) AND
12600             (GridZoneNumber = 34)

```

```

12700         THEN IF Longitude >= 21.0*(Pi/180.0)
12800
12900             THEN GridZoneNumber := 35
13000             ELSE GridZoneNumber := 33;
13100     {
13200         IF (GridZoneIndex = 22) AND
13300             (GridZoneNumber = 36)
13400             THEN IF Longitude >= 33.0*(Pi/180.0)
13500                 THEN GridZoneNumber := 37
13600                 ELSE GridZoneNumber := 35;
13700     {
13800         CenterMeridian := (6*GridZoneNumber-183)*(Pi/180.0)
13900     {
14000 end; { of PROCEDURE BTGridZoneGeneration
14100 {
14200 END. { of MODULE BTGZDG

```

```

100  (
200  PROGRAM DRVGZGR ( INPUT,OUTPUT );
300  (
400  ( PROGRAM DRVGZGR provides a test driver capability for testing )
500  ( GridZoneGeneration procedures for GUARDRAIL. )
700  (
800  TYPE
900      DegreesReal          = Real;
1000     DegreesInteger      = INTEGER;
1100     ZoneRange           = 1..60;
1200     Letters              = 'A'..'Z';
1300  VAR
1400     Longitude             : DegreesReal;
1500     Latitude              : DegreesReal;
1600     GridZoneNumber        : ZoneRange;
1700     GridZoneLetter        : Letters;
1750     CenterMeridian       : DegreesInteger;
1800  (
1900  PROCEDURE GRGridZoneGeneration
2000      (OUT) Longitude      : DegreesReal;
2100      (OUT) Latitude      : DegreesReal;
2200      (IN ) VAR GridZoneNumber : ZoneRange;
2300      (IN ) VAR GridZoneLetter : Letters;
2400      (IN ) VAR CenterMeridian : DegreesInteger); EXTERN;
2500  (
2603  ( PROCEDURE GRGridZoneGeneration models the GUARDRAIL conversion )
2604  ( of geographic coordinates to Universal Transverse Mercator )
2606  ( (UTM) coordinates grid zone designator number, letter and the )
2608  ( central meridian of the rectangle. )
2900  (
3200  BEGIN
3300  (
3400      WRITELN ( ' ENTER Longitude' );
3500      READLN ( Longitude );
3600      WRITELN ( ' ENTER Latitude' );
3700      READLN ( Latitude );
3800      GRGridZoneGeneration ( (OUT) Longitude, Latitude,
3900                          (IN ) GridZoneNumber, GridZoneLetter,
3950                          CenterMeridian );
4000
4100      WRITELN;
4200      WRITELN ( ' GridZoneNumber is ', GridZoneNumber );
4300      WRITELN;
4400      WRITELN ( ' GridZoneLetter is ', GridZoneLetter );
4500      WRITELN;
4550      WRITELN ( ' CenterMeridian is ', CenterMeridian );
4600  (
END. ( of PROGRAM DRVGZGR )

```

```

100  {
200  MODULE GRGZDG ( INPUT,OUTPUT );
300  {
400  TYPE
500      DegreesReal      = REAL;
600      DegreesInteger   = INTEGER;
700      ZoneRange        = 1..60;
800      Letters           = 'A'..'Z';
900  {
1000 PROCEDURE GRGridZoneGeneration
1100      ( (IN ) Longitude      : DegreesReal;
1200        (IN ) Latitude       : DegreesReal;
1300        (OUT) VAR GridZoneNumber : ZoneRange;
1400        (OUT) VAR GridZoneLetter : Letters;
1500        (OUT) VAR CenterMeridian : DegreesInteger );
1600 {
1700 { PROCEDURE GRGridZoneGeneration models the GUARDRAIL conversion
1800 { of geographic coordinates to Universal Transverse Mercator
1900 { (UTM) coordinates - grid zone designator numbers, letters, and
2000 { the central meridian.
2100 {
2200 { PROCEDURE GRGridZoneGeneration is referenced by:
2300 {     PROGRAM DRVGZGR
2400 {
2500 { PROCEDURE GRGridZoneGeneration makes no references.
2600 {
2700 { This procedure DOES NOT perform any data validity checks
2800 { that are not explicitly specified in the algorithm
2900 { description. This is to allow the algorithm features to be
3000 { represented more clearly.
3100 {
3200 TYPE
3300     Letters      = 'A'..'Z';
3400     IndexRange   = 1..24;
3500 {
3600 VAR
3700     GridZoneLtrList : ARRAY[1..24] OF LETTERS;
3800     GridZoneIndex    : IndexRange;
3900 {
4000 BEGIN
4100 { Initialize allowable characters array
4200 {
4300     GridZoneLtrList [1]    := 'A';
4400     GridZoneLtrList [2]    := 'B';
4500     GridZoneLtrList [3]    := 'C';
4600     GridZoneLtrList [4]    := 'D';
4700     GridZoneLtrList [5]    := 'E';
4800     GridZoneLtrList [6]    := 'F';
4900     GridZoneLtrList [7]    := 'G';
5000     GridZoneLtrList [8]    := 'H';
5100     GridZoneLtrList [9]    := 'J';
5200     GridZoneLtrList [10]   := 'K';
5300     GridZoneLtrList [11]   := 'L';
5400     GridZoneLtrList [12]   := 'M';
5500     GridZoneLtrList [13]   := 'N';
5600     GridZoneLtrList [14]   := 'P';
5700     GridZoneLtrList [15]   := 'R';
5800     GridZoneLtrList [16]   := 'R';
5900     GridZoneLtrList [17]   := 'S';
6000     GridZoneLtrList [18]   := 'T';
6100     GridZoneLtrList [19]   := 'U';
6200     GridZoneLtrList [20]   := 'V';
6300     GridZoneLtrList [21]   := 'W';

```

```

6400      GridZoneLtrList [22]      := 'X';
6500      GridZoneLtrList [23]      := 'Y';
6600      GridZoneLtrList [24]      := 'Z';
6700  {
6800      GridZoneNumber := TRUNC (31.0+(Longitude/6.0));
6900      GridZoneIndex  := TRUNC (13.0+(Latitude/8.0));
7000      GridZoneLetter := GridZoneLtrList[GridZoneIndex];
7100      CenterMeridian := 6*GridZoneNumber-183
7200  {
7300  END; { of PROCEDURE GRGridZoneGeneration
7400  {
7500  END. { of MODULE GRGZDG

```

END

DATE
FILMED

7-83

DTIC

Cont

AD-A129 182

US ARMY INTELLIGENCE CENTER AND SCHOOL USAICS SOFTWARE
ANALYSIS AND MANAG. (U) JET PROPULSION LAB PASADENA CA
P BABBY ET AL. 09 JUL 82 JPL-D-181 NAS7-918

2/2

UNCLASSIFIED

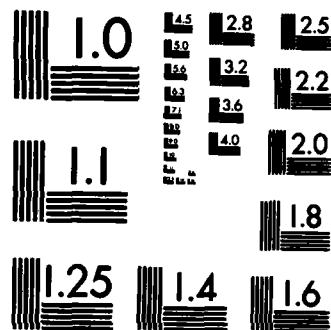
F/G 8/2

NL

END

FILED

DATE



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

SUPPLEMENTARY

INFORMATION

Connected DD 1473

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD-A129182

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ~~*****~~ D-181	2. GOVT ACCESSION NO. AD A129 182	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Analysis of Geographic Transformation Algorithms		5. TYPE OF REPORT & PERIOD COVERED FINAL
		6. PERFORMING ORG. REPORT NUMBER D-181
7. AUTHOR(s) J. Gillis, J. Radbill, A. Griesel, N. Palmer, P. Babby		8. CONTRACT OR GRANT NUMBER(s) NAS7-918
9. PERFORMING ORGANIZATION NAME AND ADDRESS Jet Propulsion Laboratory ATTN:171-209 California Institute of Technology 4800 Oak Grove, Pasadena, CA 91109		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS RE-182 AMEND # 187
11. CONTROLLING OFFICE NAME AND ADDRESS Commander, USAICS ATTN: ATSI-CD-SF Ft. Huachuca, AZ 85613-7000		12. REPORT DATE July 9, 1982
		13. NUMBER OF PAGES 35
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (Same as Item 11)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE NONE
16. DISTRIBUTION STATEMENT (of this Report) Approved for Public Dissemination		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES Prepared by Jet Propulsion Laboratory for the US Army Intelligence Center and School's Combat Developer's Support Facility		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) GEOGRAPHIC TRANSFORMATION, COORDINATE TRANSFORMATION, ALGORITHMS, MAGIIC, BETA, GUARDRAIL, TRAILBLAZER, ALGORITHM ANALYSIS, SOFTWARE ANALYSIS, IEW SYSTEMS		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report describes the findings of JPL regarding geographic transformation algorithms used in MAGIIC, GUARDRAIL, TRAILBLAZER and BETA systems. A set of parameters is developed to characterize and catalogue intelligence system algorithms in the four systems. Individual algorithms are also analyzed to determine if they are performing their functions properly.		

END

FILMED

9-85

DTIC