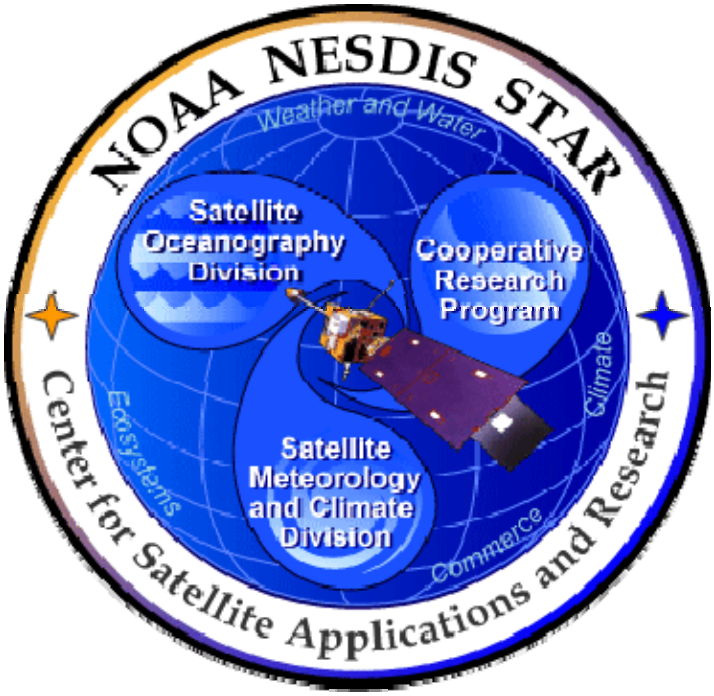

NOAA NESDIS
CENTER for SATELLITE APPLICATIONS and RESEARCH

DOCUMENT GUIDELINE

DG-8.1
DETAILED DESIGN DOCUMENT GUIDELINE
Version 3.0

TITLE: DG-8.1: DETAILED DESIGN DOCUMENT GUIDELINE VERSION 3.0

AUTHORS:

Ken Jensen (Raytheon Information Solutions)

DETAILED DESIGN DOCUMENT GUIDELINE
VERSION HISTORY SUMMARY

Version	Description	Revised Sections	Date
1.0	New Document Guideline (DG-11.1) adapted from CMMI practices and Raytheon practices by Ken Jensen (Raytheon Information Solutions)	New Document	12/29/2006
2.0	Revised by Ken Jensen (Raytheon Information Solutions) for version 2. Minor revisions to References	2	10/19/2007
3.0	Renamed DG-8.1 and revised by Ken Jensen (RIS) for version 3		10/1/2009

TABLE OF CONTENTS

LIST OF ACRONYMS

CDR	Critical Design Review
CICS	Cooperative Institute for Climate Studies
CIMSS	Cooperative Institute for Meteorological Satellite Studies
CIOSS	Cooperative Institute for Oceanographic Satellite Studies
CIRA	Cooperative Institute for Research in the Atmosphere
CL	Check List
CLI	Check List Item
CMMI	Capability Maturity Model Integration
COTS	Commercial Off The Shelf
CREST	Cooperative Remote Sensing and Technology Center
CUTR	Code Unit Test Review
DDD	Detailed Design Document
DG	Document Guideline
EPL	Enterprise Project Lifecycle
NESDIS	National Environmental Satellite, Data, and Information Service
NOAA	National Oceanic and Atmospheric Administration
PAR	Process Asset Repository
PDR	Preliminary Design Review
PG	Process Guideline
PRG	Peer Review Guideline
PRR	Project Requirements Review
RAD	Requirements Allocation Document
SG	Stakeholder Guideline
SPSRB	Satellite Products and Services Review Board
STAR	Center for Satellite Applications and Research
SWA	Software Architecture Document
TD	Training Document
TG	Task Guideline

1. INTRODUCTION

The NOAA/NESDIS Center for Satellite Applications and Research (STAR) develops a diverse spectrum of complex, often interrelated, environmental algorithms and software systems. These systems are developed through extensive research programs, and transitioned from research to operations when a sufficient level of maturity and end-user acceptance is achieved. Progress is often iterative, with subsequent deliveries providing additional robustness and functionality. Development and deployment is distributed, involving STAR, the Cooperative Institutes (CICS, CIMSS, CIOSS, CIRA, CREST) distributed throughout the US, multiple support contractors, and NESDIS Operations.

NESDIS/STAR is implementing an increased level of process maturity to support the exchange of these software systems from one location or platform to another. The Detailed Design Document (DDD) is one component of this process.

1.1. Objective

The objective of this Document Guideline (DG) is to provide STAR standards for the DDD. The intended users of this DG are the personnel assigned by the Project Lead to the task of creating a DDD for the project.

1.2. The Detailed Design Document

The purpose of the DDD is to describe the product design at a level of detail that is sufficient for the development programmers to write fully functional pre-operational code. A separate DDD is produced for each software unit that is part of the product processing system. The software units are the Layer-2 elements that are defined in the system layer product software architecture, as described in the Software Architecture Document (SWA).

DDD v1r0 is the first DDD version to be created. DDD v1r0 is produced for the Critical Design Review (CDR). It should capture the detailed design as it exists at the completion of the Design Phase of the STAR EPL.

DDD v1r1 is a planned revision of the DDD that occurs after CDR. Its purpose is to provide refinement of the code detailed design description that is typically necessary as the pre-operational code is written. DDD v1r1, a Test Readiness Review (TRR) artifact, should capture the detailed design as it exists at the commencement of unit testing.

DDD v1r2 is a planned revision of the DDD that occurs after TRR. Its purpose is to provide refinement of the code detailed design description that is typically necessary as the code is

debugged and tested. DDD v1r2, a Code Test Review (CTR) artifact, should capture the detailed design as it exists at the completion of unit testing.

Additional revisions of the DDD (v1r3, etc.) may be produced if the need for unexpected code refinement arises after CTR (i.e., during system integration and system testing).

The intended target audiences are Development Programmers and Technical Reviewers. Typically, the DDD is prepared by Development Programmers, with assistance from Development Scientists and oversight by the Development Lead.

The content of the DDD overlaps to some extent with the content of the System Maintenance Manual (SMM) Appendix. It is important that the DDD developers consult with the SMM developers during to ensure that the two documents are consistent. The DDD differs from the SMM in its detailed focus on code component design and design language. Its target audience is code development programmers. The SMM, a document with operational (Satellite Products and Services Review Board - SPSRB) heritage, is targeted at operations and maintenance personnel and focuses on providing the information needed to locate all input, intermediate and output data sets of the operational product processing system, ensure their availability, and identify and correct processing errors.

The DDD should be developed as a Microsoft Word document. Upon approval, the approved version of the DDD may be converted to an Adobe pdf file for storage in the project artifact repository.

1.3. Background

This DG defines guidelines for producing a DDD. This DG has been adapted from the Capability Maturity Model Integration (CMMI) for Development V1.2 guidelines and practices (http://www.sei.cmu.edu/publications/documents/06.reports/06tr008.html). It has been tailored to fit the STAR EPL process.

1.4. Benefits

A DDD developed in accordance with the standards in this DG assists the development team to write pre-operational code that implements the product detailed design. It is therefore a requirement that a DDD be developed in accordance with the guidelines in this document before pre-operational code is approved for transition to operations.

1.5. Overview

This DG contains the following sections:

Section 1.0	-	Introduction
Section 2.0	-	References
Section 3.0	-	Standard Table of Contents
Section 4.0	-	Section Guidelines
Appendix A	-	Examples
Appendix B	-	Templates

2. REFERENCE DOCUMENTS

SWA: Software Architecture Document contains the software architecture and data flows for the project algorithm. This information will be useful for the developer of the DDD in completing Sections 2 and 3. This document will be available to approved users in the project artifact repository.

RAD: Requirements Allocation Document contains the basic and derived requirements for the work products and the initial allocation of the requirements to system components and product components. This information will be useful for the DDD developer in completing Sections 2 and 3 of the DDD. This document will be available to approved users in the project artifact repository.

All of the following references are STAR EPL process assets that are accessible in a STAR EPL Process Asset Repository (PAR) on the STAR web site:

http://www.star.nesdis.noaa.gov/star/EPL_index.php.

PG-1: STAR EPL Process Guideline provides the definitive description of the standard set of processes of the STAR EPL.

PG-1.A: STAR EPL Process Guideline Appendix, an appendix to PG-1, is a Microsoft Excel file that contains the STAR EPL process matrix (Stakeholder/Process Step matrix), listings of the process assets and standard artifacts, descriptions of process gates and reviews, and descriptions of stakeholder roles and functions.

PRG-8.1: Critical Design Review Guidelines are the guidelines for the CDR. It is useful for the developer of DDD v1.0 to understand what the reviewers will expect when reviewing the DDD.

CL-8.1: Critical Design Review Check List is the check list for the CDR. It is useful for the developer of DDD v1.0 to understand the specific Check List Items (CLI) that the reviewers of the DDD will be required to approve.

PRG-9: Test Readiness Review Guidelines are the guidelines for the TRR. It is useful for the developer of DDD v1.1 to understand what the reviewers will expect when reviewing the DDD.

CL-9: Test Readiness Review Check List is the check list for the TRR. It is useful for the developer of DDD v1.1 to understand the specific CLI that the reviewers of the DDD will be required to approve.

PRG-10: Code Test Review Guidelines are the guidelines for the CTR. It is useful for the developer of DDD v1.2 to understand what the reviewers will expect when reviewing the DDD.

CL-10: Code Test Review Check List is the check list for the CTR. It is useful for the developer of DDD v1.2 to understand the specific CLI that the reviewers of the DDD will be required to approve.

DG-0.1: STAR Document Style Guideline is a STAR EPL Document Guideline (DG) that provides STAR standards for the style and appearance of STAR documents developed as Microsoft Word files

SG-16: STAR EPL Development Programmer Guidelines provides a description of standard tasks for Development Programmers, including development of the DDD.

TG-8: STAR EPL Detailed Design Task Guidelines provides a description of standard tasks for process step 8, during which DDD v1.0 is developed.

TG-9: STAR EPL Code Development and Test Task Guidelines provides a description of standard tasks for process step 9, during which DDD v1.1 is developed.

TG-10: STAR EPL Code Test and Refinement Task Guidelines provides a description of standard tasks for process step 10, during which DDD v1.2 is developed.

3. STANDARD TABLE OF CONTENTS

4. SECTION GUIDELINES

This section contains the STAR guidelines for each section of the DDD.

The DDD should follow the STAR standard for style and appearance, as stated in DG-0.1.

4.1. Table of Contents

The Table of Contents can be inserted by using Word's Insert → Reference → Index and Tables → Table of Contents function or by pasting the Table of Contents from this DG into your document and updating it for the section headers you make for your document. Use a page break if necessary to ensure that the Table of Contents appears at the top of a page.

4.2. List of Figures

A List of Figures should be provided after the Table of Contents. A page break should be used if necessary to ensure that the List of Figures appears at the top of a page. To create a List of Figures, use Word's Insert → Reference → Index and Tables → Table of Figures function, selecting the "Table of Figures" Style. Alternatively, the List of Figures can be created by pasting the List of Figures for this DG into your document.

Figures should be created by using Word's Insert → Picture → From File function or Word's Insert → Object function. Figures should be numbered X.Y, where X is the main section number where the figure resides and Y = 1,N is the ordered number of the figure in the section. Figure captions should have Arial bold 12 point font, should be center justified, and should have a "Table of Figures" Style. A Figure Caption template is provided in Appendix B of this DG.

4.3. List of Tables

A List of Tables should be provided after the List of Figures. The List of Tables can appear on the same page as the List of Figures, with three blank lines separating them, provided both lists can fit on the same page. If both lists cannot fit on the same page, a page break should be used to ensure that the List of Tables appears at the top of a page.

To create a List of Tables, use Word's Insert → Reference → Index and Tables → Table of Figures function, selecting the "Table - Header" Style. Alternatively, the List of Tables can be created by pasting the List of Tables for this DG into your document.

Tables should be created with the Table → Insert → Table function. Tables should be numbered X.Y, where X is the main section number where the table resides and Y = 1,N is the ordered number of the table in the section. Table titles should have Arial bold 12 point font, should be center justified, and should have a "Table - Header" Style. A Table Title template is provided in Appendix B of this DG. Table text should have Arial regular 10 point font.

4.4. List of Acronyms

The use of acronyms is encouraged. A two word or longer name for an item (e.g., Research Project Plan) should be given an acronym (e.g., RPP) if the name is used more than once in the document. A List of Acronyms should be provided after the List of Tables. The List of Acronyms should be in alphanumeric order. Use the List of Acronyms in this DG as a template. A page break should be used if necessary to ensure that the List of Acronyms appears at the top of a page.

4.5. Section 1 – Introduction

The DDD shall include an Introduction Section. This section shall include

- A well-defined purpose and function for the document
- A brief overview of the contents of each main section
- A revision/change history, focusing on updates since the previous revision

The standard purpose and function of the DDD is to provide a method of detailing the Unit-Layer structural procedure within the unit, place these in the context of the product system data processing chain, and specify the data, its formats, and the relationships that exist between the data to form the basis for the detailed design of the unit code.

4.6. Section 2 – Unit Description

Provide a description of the unit at a level of detail sufficient for development programmers to be able to write pre-operational unit code and for design reviewers to understand its function, capabilities and limitations to be able to verify that the requirements allocated to the unit can be met. Subsections should include Purpose and Function, Requirements Allocation, Interfaces, Sub-Units, Functional Sequence, Design Language, Assumptions and Limitations.

The subsection for Purpose and Function should explain the role of the unit in the product processing system, the major functional steps, and how these steps satisfy the purpose of the unit. The content should be primarily textual. References to appropriate figures and tables in the SWA should be made.

The subsection for Requirements Allocation should identify which requirements have been allocated to the unit, noting whether the unit provides a complete or a partial satisfaction of the requirement. Note any revisions to the requirements that have been driven by design constraints for this unit. Refer to the RAD and draw material from the RAD as warranted. Refer to Section 4 of the SWA, where this information is tabulated.

The subsection for Interfaces should list the external interfaces to the unit, including interfaces with other units in the product processing system. There should be a description of each interface. This description should include the information needed by a development programmer for writing the needed I/O code. References to appropriate figures and tables in the SWA should be made. Ensure consistency with the SWA.

The subsection for Sub-Units should consist of a list of the sub-units for the unit. The software sub-units are the Layer-3 elements that are defined in the System-Layer product software architecture, as described in the SWA. These elements should be numbered in the SWA. Identify each sub-unit with the correct number from the SWA. References to the appropriate figures and tables in the SWA should be made.

The subsection for Functional Sequence should place the major functions of the unit in sequential order, with external and internal data flows noted. The material in this subsection should be obtained from the unit's data flow diagram and data flow table, as documented in the SWA. References to the appropriate figures and tables in the SWA should be made.

The subsection for Design Language should consist of an exposition of the unit data flows and functional sequence in a style that resembles code. The idea is to use the software architecture to begin to visualize how the code will look for the unit's main program. This subsection should be a collaboration between the code designer and the programmer. It is here that the design begins to be translated into a "pseudo-code" in a way that is understandable by both the designer and the programmer. In this way, the designer verifies

that the programmer understands how to implement the design and the programmer is assured that the pseudo-code is a satisfactory basis for programming.

The subsection for Maintenance should describe how the software unit code and its interfaces are expected to be maintained. Include a maintenance plan for each non-developmental (Commercial Off The Shelf (COTS), government off the shelf, and reuse) item at the Unit-Layer.

The subsection for Assumptions and Limitations should state all assumptions that were made in determining that the processing system as designed will meet the requirements. Assumptions can be on the quality, latency, availability of inputs, maintainability, and the capabilities of the processing environment. If an assumption may be questionable, note this as a risk. State all identified limitations that may impact on the system's ability to meet requirements. If a limitation poses a requirements risk, note this.

4.7. Section 3 – Sub-Unit Descriptions

Provide a description of each of the unit's sub-units at a level of detail sufficient for development programmers to be able to write pre-operational unit code and for design reviewers to understand its function, capabilities and limitations to be able to verify that the requirements allocated to the unit can be met. There are separate subsections for each sub-unit. Each sub-unit subsection should contain its own subsections. Sub-unit subsections should include Purpose and Function, Requirements Allocation, Interfaces, Functional Sequence, Design Language, Assumptions and Limitations. Typically, each sub-unit will be a self-contained subprogram of the unit program. The sub-unit subsections are very similar to the unit subsections of Section 2, applied to the relevant subprogram as opposed to the main unit program.

The subsection for Purpose and Function should explain the role of the sub-unit in the product processing system, the major functional steps, and how these steps satisfy the purpose of the unit. The content should be primarily textual. References to appropriate figures and tables in the SWA should be made.

The subsection for Requirements Allocation should identify which requirements have been allocated to the sub-unit, noting whether the sub-unit provides a complete or a partial satisfaction of the requirement. Note any revisions to the requirements that have been driven by design constraints for this sub-unit. Refer to the RAD and draw material from the RAD as warranted. Refer to Section 4 of the SWA, where this information is tabulated.

The subsection for Interfaces should list the external interfaces to the sub-unit, including interfaces with other units in the product processing system and with the other sub-units of

this unit. There should be a description of each interface. This description should include the information needed by a development programmer for writing the needed I/O code. References to appropriate figures and tables in the SWA should be made. Ensure consistency with the SWA.

The subsection for Functional Sequence should place the major functions of the sub-unit in sequential order, with external and internal data flows noted. The material in this subsection should be obtained from the sub-unit's data flow diagram and data flow table, as documented in the SWA. References to the appropriate figures and tables in the SWA should be made.

The subsection for Design Language should consist of an exposition of the sub-unit data flows and functional sequence in a style that resembles code. The idea is to use the software architecture to begin to visualize how the code will look for the relevant unit subprogram. This subsection should be a collaboration between the code designer and the programmer, It is here that the design begins to be translated into a "pseudo-code" in a way that is understandable by both the designer and the programmer. In this way, the designer verifies that the programmer understands how to implement the design and the programmer is assured that the pseudo-code is a satisfactory basis for programming.

The subsection for Maintenance should describe how the software sub-unit code and its interfaces are expected to be maintained. Include a maintenance plan for each non-developmental (COTS, government off the shelf, and reuse) item at the Unit-Layer.

The subsection for Assumptions and Limitations should state all assumptions that were made in determining that the processing system as designed will meet the requirements. Assumptions can be on the quality, latency, availability of inputs, maintainability, and the capabilities of the processing environment. If an assumption may be questionable, note this as a risk. State all identified limitations that may impact on the system's ability to meet requirements. If a limitation poses a requirements risk, note this.

4.8. Section 4 – File Descriptions

Every input, output, and intermediate file in the system architecture should be fully described, including control files, parameter files, look up tables, input data files, ancillary data files, intermediate data files, and output data files. Each of these types of files should be described in its own subsection.

Control files are typically scripts that define run control parameters. For each file, indicate the variables it contains, the file format, and how the values of the variables are read into

the program or subprograms that use them. The file format is usually best described by a table.

Parameter files contain the values of variables that are fed into the unit program or sub-programs. For each file, indicate the variables it contains, the file format, and how the values of the variables are read into the program or subprograms that use them. The file format is usually best described by a table.

Look up tables typically contain the values of variables binned by a range of conditions, or stratifications. For each file, indicate the variables it contains, how they are binned, the file format, and how the values of the variables are read into the program or subprograms that use them. The file format is usually best described by a table.

For each ancillary data file, indicate the source of the file, how the file is obtained, references to relevant file description documentation by the file provider, the variables contained in the file, how they are binned (if they are binned), the file format, and how the values of the variables are read into the program or subprograms that use them. The file format is usually best described by a table.

For each input data file, indicate the variables it contains, the file format, and how the values of the variables are read into the program or subprograms that use them. The file format is usually best described by a table.

For each intermediate data file, indicate the variables it contains, the file format, and how the values of the variables are read into the program or subprograms that use them. The file format is usually best described by a table.

For each output data file, indicate the variables it contains, the file format, and how the values of the variables are read into the program or subprograms that use them. The file format is usually best described by a table.

4.9. Section 5 – List Of References

This section should consist of a List of References that includes all references cited in the document. Include all references deemed useful by the Product Team. References should be listed in alphabetical order. References that begin with an author list should begin with the last name of the lead author. A template is provided in Appendix B.

APPENDIX A - EXAMPLES

Examples of DDDs that follow the STAR standards and guidelines will be developed and placed in the STAR EPL PAR.

APPENDIX B - TEMPLATES

This appendix contains templates for specific pages and sections of the DDD.

B.1 Cover Page Template:

In this template, <X> = 1.0 for version 1, <X> = 1.1 for version 1 revision 1, <X> = 2.0 for version 2 etc. <Project Name> should be the actual approved name of the Project, and <Unit Name> should be the name of the software unit, as defined in the Project SWA.

NOAA NESDIS
CENTER for SATELLITE APPLICATIONS and RESEARCH

<PROJECT NAME>
<UNIT NAME> UNIT
DETAILED DESIGN DOCUMENT

Version <X>

B.2 Document Header Template:

In this template, <X> = 1.0 for version 1, <X> = 1.1 for version 1 revision 1, <X> = 2.0 for version 2 etc.

In this template, <Unit Name> should be the name of the software unit, as defined in the Project SWA.

In this template, <Project Name> should be the actual approved name of the Project.

In this template, <Y> = the actual page number.

In this template, <Z> = the actual total number of pages

B.3 Document Cover Page Footer Template:

B.4 Document Footer Template:

Hardcopy Uncontrolled

B.5 Approval Page Template:

In this template, <X> = 1.0 for version 1, <X> = 1.1 for version 1 revision 1, <X> = 2.0 for version 2 etc. <Project Name> should be the actual approved name of the Project, and <Unit Name> should be the name of the software unit as defined in the project's SWA.

TITLE: <PROJECT NAME> <UNIT NAME> UNIT DETAILED DESIGN DOCUMENT VERSION <X>

AUTHORS:

<Lead Author>

<Co-Author 1>

<Co-Author 2>

<etc.>

APPROVAL SIGNATURES:

_____ <Actual Signature Date>
<Name of Project Development Lead> Date
Project Development Lead

_____ <Actual Signature Date>
<Name of Project Manager> Date
Project Manager

_____ <Actual Signature Date>
<Name of Agency Approver> Date
Agency

B.6 Version History Page Template:

In this template, <Project Name> should be the actual approved name of the Project, and <Unit Name> should be the name of the software unit as defined in the project's SWA.

<PROJECT NAME>
<UNIT NAME> UNIT
DETAILED DESIGN DOCUMENT
VERSION HISTORY SUMMARY

Version	Description	Revised Sections	Date
1.0	There is no version 1. The original version is 2.0 to synchronize it with the SWA version	N/A	N/A
2.0	Created by <Name of Developer(s)> of <Name of Developers' Agency/Company>.	New Document	<Actual date of Latest approval signature>
2.1	Revised by <Name of Developer(s)> of <Name of Developers' Agency/Company> to correct defects found during Critical Design Review	<applicable sections>	<Actual date of Latest approval signature>
2.2	[As needed] Revised by <Name of Developer(s)> of <Name of Developers' Agency/Company> to correct defects found during Critical Design Review	<applicable sections>	<Actual date of Latest approval signature>
etc.			

B.7 Figure Caption Template:

Figure 2.3 - <Figure caption in Arial regular 12 point font>

B.8 Table Title Template:

Table 4.5 - <Table title in Arial regular 12 point font>

B.9 List of References Template:

Ackerman, S. *et al.* (1997). Discriminating clear-sky from cloud with MODIS: Algorithm Theoretical Basis Document, Version 3.2.

Asrar, G., M. Fuchs, E. T. Kanemasu, and J. L. Hatfield (1984). Estimating absorbed photosynthetically active radiation and leaf area index from spectral reflectance in wheat. *Agron. J.*, 76:300-306.

Bauer, E., and Kohavi, R., (1998). An empirical comparison of voting classification algorithms: bagging, boosting, and variants, *Machine Learning*, **5**: 1-38.

Bonan, G.B. (1995). Land-atmosphere interactions for climate system models: Coupling biophysical, biogeochemical, and ecosystem dynamical processes. *Remote Sens. Environ.*, 51:57-73.

Food and Agriculture Organization of the United Nations*, Digital Soil Map of the World and Derived Soil Properties-Version 3.5*, FAO/UNESCO, Rome, 1995.

Friedl, M. A., and C.E. Brodley (1997). Decision tree classification of land cover from remotely sensed data. *Remote Sens. Environ.*, 61:399-409.

Scepan, J. (1999), Thematic validation of high-resolution global land-cover data sets. *Photogramm. Eng. Remote Sens.*, 65:1051-1060.

Shukla, J., C. Nobre, and P. Sellers (1990). Amazon deforestation and climate change. *Science*, 247:1322-1325.

Wilson, M.F., and A. Henderson-Sellers (1985). A global archive of land cover and soils data for use in general circulation models. *J. Clim*., 5:119-143.

Wu, A., Z. Li, and J. Cihlar (1995). Effects of land cover type and greenness on advanced very high resolution radiometer bidirectional reflectances: analysis and removal. *J. Geophys. Res.*, 100: 9179-9192.

END OF DOCUMENT