TROPICAL RAINFALL MEASURING MISSION
SCIENCE DATA AND INFORMATION SYSTEM

Interface Control Specification Between the Tropical Rainfall Measuring Mission Science Data and Information System (TSDIS) and the TSDIS Science Users (TSU)

TSDIS-P907

Volume 1:
Algorithm Software Development and Delivery

Release 5.01

Prepared for:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
GODDARD SPACE FLIGHT CENTER
Code 902
Greenbelt, Maryland 20771

June 2, 2000

TABLE OF CONTENTS

TABLE

LIST OF APPENDICES

1. INTRODUCTION

1.1 IDENTIFICATION OF DOCUMENT

This is the Interface Control Specification (ICS) for the Tropical Rainfall Measuring Mission (TRMM) Science Data and Information System (TSDIS) Science Users' (TSU) interface with the TSDIS.

1.2 SCOPE OF DOCUMENT

This document will be limited to addressing the TSDIS Science User algorithms submitted for inclusion in the TSDIS. The scope of this ICS is limited to the effort which defines the interfaces required for the algorithm integration into TSDIS. It does not address the algorithm development efforts themselves, nor does it address TSDIS software development efforts. The ICS is divided into five volumes. Volume 1 - Algorithm Software Development and Delivery defines and describes the various software interfaces required for science algorithm integration into TSDIS. Volume 2 - Toolkit User's Guide provides the programmer user guide for the TSDIS toolkit routines. Volume 3 - File Specifications - Level 1 provides the data format and content for all Level 1 TSDIS data products. Volume 4 - File Specifications - Levels 2 & 3 provides the data format and content for all Level 2 and Level 3 TSDIS data products. Volume 5 - File Specifications - Browse provides the data format and content for all TSDIS Browse products. Metadata definitions for products are included in these file specification volumes. Other types of metadata will be included in the TSDIS Software Design Specification.

1.3 PURPOSE AND OBJECTIVES OF DOCUMENT

The term "ICS", when used without a volume number, refers to the entire suite of volumes (Volumes 1 through 5). This TSDIS-TSU ICS describes and defines the various software interfaces required for the integration of the TSU's algorithm code into the TSDIS environment. The ICS (Volumes 1 through 5) constitutes a formal understanding between the TSUs and the TSDIS developers regarding algorithm integration into TSDIS. This document will not be altered without approval by the TSDIS Configuration Control Board.

This document is intended for release during Version 3 (V3) algorithm deliveries (scheduled for 3/96 through 10/96) to provide guidance to the science algorithm developers. The V3 algorithms constitute 90% launch ready versions of the algorithms. (Version 4 (V4), the launch ready version, is scheduled to allow for fine tuning of the algorithms prior to launch).

The objective of this ICS is to define the specific software interfaces for science algorithms to enable integration of the algorithms into the TSDIS environment. Specific objectives of this ICS are as follows.

a) To define data formats and programming practices which are necessary for integration of science algorithm code into the TSDIS.
b) To define software toolkit formats and structures.

c) To promote compatibility with Earth Observing System Data and Information System (EOSDIS) by providing a means for establishing consistency among the algorithms as well as data consistency within TSDIS which are in accordance with EOSDIS requirements.

1.4 DOCUMENT STATUS AND SCHEDULE

The first release (R1) of the ICS was completed in February 1995. The R1 version of the ICS provided information on the TSDIS environment and anticipated toolkits. The second release (R2) of the ICS was completed in August 1995. The R2 version included Volumes 1, 2, 3, and 4. The third release (R3) of the ICS is scheduled for July 1996. The R3 version will include volume 5 as well as updates to Volumes 1 through 4.

Subsequent changes to the document may occur when the Science Team and the TSDIS Team evaluate changes required by the TSU's algorithm programs, or any changes to the TSDIS requirements. The teams will jointly determine the appropriate changes required for this ICS. New releases of the document will be subject to the TSDIS Configuration Control. Approved (signed-off) versions of this ICS will be disseminated to appropriate TRMM community members.

The current anticipated schedule for the ICS releases is as follows:

Release 1	February 1995 - completed
Release 2	August 1995 - completed
Release 3	July 1996 - completed; any further changes will be handled through the TSDIS Configuration Change Board
Release 4	September 1998 - completed
Release 5	October 1999 - completed

Questions regarding the content of this document should be directed to Dr. Michael McCumber Code 902, NASA/GSFC.

1.5 DOCUMENT ORGANIZATION

Volume 1 of the ICS is organized into the following sections.

Section 1.0 INTRODUCTION - This section presents the information regarding this document in terms of document identification, scope, purpose, status, and schedule. It further defines the contents of the document.

Section 2.0 RELATED DOCUMENTATION - This section presents the documentation used in the preparation of this document which includes parent documents, applicable documents, and information documents.

Section 3.0 SOFTWARE TOOLKIT - This section defines the software toolkit interface conventions and the types of software toolkits which TSDIS will make available to the algorithm developers.

Section 4.0 DEVELOPMENT OF ALGORITHMS - This section defines the type of programming practices requested of each TSU to facilitate integration of algorithm programs within TSDIS. This section defines the processing parameters to be identified for each algorithm (included in the algorithm submittal package), file naming conventions, data formats, standards, conventions, and representations required for integration into TSDIS. This section defines the programming constraints required for source code compatibility with the TSDIS operating environment. This includes a variety of computer environment issues such as programming languages supported, compilers supported, and software support library compatibility. This section also describes the TSDIS algorithm test environment.

Section 5.0 SYNTHETIC DATA SETS - This section defines the synthetic TRMM data sets and their availability in support of the Version 2, Version 3, and Version 4 algorithm deliveries.

Section 6.0 PRODUCT GENERATION - This section defines the interactions with the TSUs for product generation on a daily basis.

Section 7.0 ABBREVIATIONS AND ACRONYMS - This section lists the abbreviations and acronyms used in this document.

Section 8.0 GLOSSARY - This section defines the unique terminology used in this document.

Section 9.0 APPENDICES - This section contains the appendices for this document.

APPENDIX A ALGORITHM SUBMITTAL PACKAGE CONTENTS - This appendix gives a list and brief description of the contents of the algorithm delivery package to be provided by the TSU. This includes a skeletal outline of a typical algorithm User's Guide to be submitted by each TSU to TSDIS as part of the algorithm delivery package.

APPENDIX B RECOMMENDED PROGRAMMING PRACTICES FOR ALGORITHM DEVELOPERS. This appendix provides suggested programming practices to be followed by TSUs that will aid in the integration and maintenance of the algorithm programs.

2. RELATED DOCUMENTATION

2.1 PARENT DOCUMENTS

The following document serves as the parent document to this ICS.

1) Interface Control Plan (ICP) Between the Tropical Rainfall Measuring Mission Science Data and Information System (TSDIS) and the TSDIS Science User (TSU), TSDIS-P907, May 22, 1995

2.2 APPLICABLE DOCUMENTS

This document adheres to NASA-DID-999 as found in the NASA Software Documentation Standard Software Engineering Program NASA-STD-2100-91 dated July 29,1991. The following documents are applicable to the contents of this ICS.

2) Tropical Rainfall Measuring Mission Science Data and Information System (TSDIS) Project Management Plan, TSDIS-M000, August, 1994
3) TRMM Science Requirements Document, August 30, 1995
4) Appendix B: Memorandum of Understanding (MOU) between the Tropical Rainfall Measuring Mission (TRMM) Project and the EOS Ground System and Operations Project (GSOP) for Science Data Archive and Distribution Support, TRMM-490-003, October 1991
5) TSDIS Requirements Document Version 5, TSDIS-P200, October 20, 1995

2.3 INFORMATION DOCUMENTS

The following documents provide reference material used in the development of this ICS.

6) TRMM Science Operations Concept and Data Processing Scenarios Document (Revision 1), September 1991
7) Proposed ECS Core Metadata Standard Release 2.0, 420-TP-001-005, December 1994
8) EOSDIS Version 0 Data Product Implementation Guidelines, GSFC 50-003-04, GSFC, March 1, 1994
9) TSDIS Phase I Prototype Final Report and Lessons Learned, GSFC 902.3, February 23, 1994.
10) TSDIS Phase II Prototype Lessons Learned, TSDIS-R006.2-V1, GSFC 902.3, August 1994.
11) Flight Dynamics Division (FDD) Generic Data Product Formats Interface Control Document, June 1991
12) TRMM VIRS CDRL 303, Engineering Telemetry Description Final, January 1994
13) TRMM TMI CDRL 507, TMI Technical Description Document, March 1993

14) TRMM TMI CDRL 303, Engineering Telemetry Description Preliminary, February 1993

15) TRMM Housekeeping Packet Definitions, a presentation by Bruce Love, February 1994

16) Interface Control Document Between the Sensor Data Processing Facility and the TRMM Customers, Draft, February 8, 1995

17) User’s Manual SSM/I Antenna Temperature Tapes, Remote Sensing Systems, Frank J. Wentz, March 1988

18) Getting Started with HDF, Draft, Version 3.2, May 1993, University of Illinois at Urbana-Champaign

19) TRMM Telemetry and Command Handbook, TRMM-490-137, April 1995

20) Planetary Data System Standards Reference, Jet Propulsion Laboratory, Version 3.2, July 24, 1995 (http://stardust.jpl.nasa.gov/stdref)

21) Interface Control Document Between EOSDIS Core System (ECS) and TRMM Science Data and Information System (TSDIS) for the ECS Project, 209-CD-007-003, November 1995, ECS Project, Hughes Information Technology Corporation

22) Science Data Processing Segment (SDPS) Database Design and Database Schema Specifications for the ECS Project, 311-CD-002-004, December 1995, ECS Project, Hughes Information Technology Systems

23) MIPSpro 64-Bit Porting and Transition Guide, 007-2391-002, 1994-1996, Silicon Graphics, Inc. (http://www.sgi.com/Technology/Tech Pubs/lib/librarian.cgi)

3. SOFTWARE TOOLKITS

3.1 SOFTWARE TOOLKIT APPROACH

A science algorithm software developers' Toolkit provides seamless integration of science algorithm software into the TSDIS environment. The science algorithm software modules are developed on computing environments other than TSDIS (the algorithm developer's home computing environment) and then integrated into the TSDIS Integration and Test Environment (ITE) and operational production environment. The Toolkit will minimize the differences between different environments and permit the same source code to be compiled and executed on each environment without modification.

The Toolkit will consist of a library with functions. Internal details of the functions for the various environments may differ, but the Application Program Interface (API) will be identical in each environment. A call to the same function, with the same parameters in different environments, will produce the same results even though the internal code may be different.
Besides masking the differences between environments, the Toolkit will provide functions which isolate software developers from manipulation of TSDIS system resources. This will preclude the need for developers to know how to manipulate these TSDIS-unique resources consistently and safely. The system resources include disk file I/O, and error/status logging and reporting.

The Toolkit will also provide a single source of commonly used functions. Science algorithm software will use many mathematical, scientific, and engineering functions which can be put into shared libraries. Commonly used functions will be provided by IMSL (COTS) and TSDIS (e.g., geolocation). Use of these libraries will enhance consistency of results from algorithm to algorithm. The following is a list of categories of toolkits which will be made available to algorithm developers. Volume 2 of the ICS - Toolkit User's Guide - will provide the details of the specific routines and their calling sequences.

Table 3.1-1
Toolkit Categories

Toolkit Category Name	**Function**
Input/Output	Perform reads and writes of data and metadata
Mathematical	General mathematical routines
Conversions	Provides constant and unit conversions
Geo-location	Perform geo-location of pixels
Error Handling	General error processing

4. DEVELOPMENT OF ALGORITHMS

4.1 LANGUAGES AND COMPILERS

The TSDIS will support the ANSI FORTRAN 77, and ANSI C. The following extensions to FORTRAN-77 will be supported:

a) INCLUDE statements
b) DO WHILE controlling loop
c) ENDDO statement for termination of DO loop
d) IMPLICIT NONE statement
e) BYTE data type
f) STRUCTURE data type
g) RECORD data type
h) names up to 31 characters in length
i) extended character set that includes lower case letters, underscore, left and right angle brace, quotation mark, percent sign, and ampersand
j) in-line comments (exclamation point (!) may be used to mark beginning of a comment at the end of the line containing data declaration or executable)
k) blank lines

TSUs using other extensions to these languages or other forms of FORTRAN or C will need to update their source code to conform to the ANSI standards (with indicated extensions) so that their algorithm can be integrated into TSDIS.

4.2 OPERATING SYSTEMS

The TSDIS operating system for both the production system and the ITE is the IRIX V6.2 operating system with full 64-bit implementation residing on a Silicon Graphics (SGI) Challenge L (R 10,000 CPU) platform. The same operating system is used for the TRMM Office Ground Validation SGI Onyx (R 4400 CPU) platform.

4.3 SUPPORT ENVIRONMENT

This section describes the various components of the TSDIS software support environment that will be available for the integration of TSU software into TSDIS.

4.3.1 Software Libraries

TSUs who need special software libraries should make their requirements known to TSDIS as soon as possible (but no less than 3 months before algorithm submittal to TSDIS) to be certain that their requirements will be met. This includes any TSU-defined software libraries.

Currently, the International Mathematical and Statistics Library (IMSL) software library has been selected.

4.3.2 Debuggers

The porting of a TSU's algorithm code into the TSDIS environment may necessitate the use of debugging tools to determine the cause of any porting problems encountered. The TSDIS environment will have the debugging tools available for the ITE. The specific debugger is SGI’s CASEVision. CASEVision is a Computer Aided Software Engineering (CASE) tool used to aid in debugging and profiling of source code. CASEVision is also available on the pre-operational ITE.

4.4 TEST ENVIRONMENT

TSDIS will make an ITE available for testing the TSU’s V3 and V4 algorithms. TSDIS will use the pre-operational ITE to ensure that the TSU’s V3 and V4 algorithms execute properly in the TSDIS environment. The TSDIS pre-operational ITE will be used for stand-alone testing of the V2 algorithms. Currently, the pre-operational ITE uses an SGI workstation using an IRIX operating system. The SGI workstation supports C, FORTRAN 77, MOTIF, and X-windows.

The operational Integration and Test Environment, which will be available by the fall of 1996, will be used to integrate all algorithms beginning with Version 3 into TSDIS. The operational ITE will also be used to test any post launch science algorithms. This process is described in the TSDIS-TSU ICP.

4.5 CONSTRAINTS

4.5.1 Data Handling

Implicit initial data values (defaults) for variables in C and Fortran code are not common across different computing systems. For example, some initial values may be zero (0) while others may be negative one (-1). To avoid problems with initial data values, TSUs are requested to develop their code using explicit default values for all data items. This can prevent abnormal termination once testing begins in TSDIS.

4.5.2 Quality Control

An indication of the quality of the ground validation (GV) data produced by TSDIS is necessary. All GV data products must have a quality indicator which will be part of the metadata for the product (see the file specification volumes for further definitions of quality control indications). Each GV product will have an overall quality check (QC) flag which the TSU will set after the product has been generated. This will be accomplished via the RST. TSDIS will be responsible for transferring these QC indicators to EOSDIS. For satellite products, the QC flag always has the value “NOT BEING INVESTIGATED”. Additionally, there are product QC flags which are computed during the execution of the algorithm and are a part of the product metadata. However, the values to be used for any QC flags are the responsibility of the TSUs and are, therefore, not controlled by this ICS. The TSUs are responsible for defining these quality indicators for their products.

4.5.3 Portability and Integration Practices

The following programming practices need to be followed by the TSUs to permit portability of their code to the TSDIS environment.

a) No user interaction must be required for the algorithm to execute.

b) The algorithm should not write messages to the standard output (screen or printer).

c) Shell scripts should be portable (Bourne shell is recommended).

d) Equivalence statements should not be used.

e) Bit manipulations should not be used.

f) System calls should not be used (except iarg and getarg).

g) Common blocks should not be used.

h) TSDIS would prefer that algorithms be written to execute in 64-bit mode. However, if the algorithm fails in 64-bit mode, there are two other possibilities, both of which result in slower execution. One is compiling on a 64-bit machine with the 32-bit flag; the other is compiling on a 32-bit machine and copying the executable to a 64-bit machine.

An algorithm may be made executable on a 64-bit machine by observing the porting recommendations in the MIPSpro 64-bit Porting and Transition Guide. In the C language, both long ints and pointers become 64 bits on the 64-bit machine.

4.5.4 Processing Parameter Conventions

The TSU algorithms will receive file names and other parameters as arguments on the command line. TSDIS will include these command line parameters in the scheduler call used to start the algorithm's execution. The specific parameters for each algorithm are listed in Appendix A, section A.1.3.

4.6 SUBMITTAL MECHANISM

The Algorithm Developers will submit an Algorithm Submittal Package to TSDIS when the algorithm has been approved by the Science Board (or by the TRMM Science Team during pre-launch) for integration into TSDIS. The package should include the algorithm software along with supporting documentation. The precise material to be included in the algorithm submittal package is described in Appendix A. This package can be submitted in the following ways: 1) ftp of the files containing the required information; or 2) delivery via 8 mm tape.

TSDIS will establish a directory to which the algorithm submittal packages are to be transferred. If ftp is used, the TSUs will perform a "put" of the files from the developer's machine once the Science Control Board has agreed that the algorithm should be integrated into TSDIS. If the delivery is by tape, then TSDIS personnel will upload the files into the designated directory.

Algorithm Developers who submit science algorithms which require access to external data not currently supported by TSDIS must make their data requirements known and identify sources of the required data. Interface definitions and impact on product processing will need to be assessed by TSDIS as part of the algorithm integration process.

4.7 RETURN STATUS

Algorithms should force a return status of 0 (success). The return status is passed to the scheduler, which uses this information to determine whether to proceed with normal processing or follow an error recovery process. If the algorithm is written in C, it must use “exit(0)”. If the algorithm is written in FORTRAN, it must use “CALL EXIT (0)”. If the algorithm is written as a script that calls programs, the programs must use the above returns and the return status of the program must be passed out as the return status of the script.

5. SYNTHETIC DATA SETS

5.1 DATA SETS FOR V2 ALGORITHM DELIVERY

Numerically simulated data of the TRMM Microwave Imager (TMI), Visible and Infrared Sensor (VIRS), and Precipitation Radar (PR) have been developed to provide TSUs with Level 1B synthetic data sets for algorithm development and testing. TMI algorithm developers have used Special Sensor Microwave/Imager (SSM/I) data for testing because of the similarity between the two instruments. Likewise, VIRS algorithm developers have often used Advanced Very High Resolution Radiometer (AVHRR) data. Because the PR has no direct instrument heritage, there are no existing data sets which are completely satisfactory for testing. Hence, the generation of realistic synthetic PR data has been viewed as the highest priority task in the synthetic data effort.

Level 1B synthetic data sets have been generated for the V2 algorithms on an *ad hoc* basis with algorithm developers making requests directly to the synthetic data developer. These data sets have been delivered in a binary format, and no higher level data sets have been available. The TMI and VIRS simulations execute very quickly, and multi-orbit data sets can be produced within a few hours of wall clock time. However, the PR simulation runs much slower due to the complexity of its three-dimensional radar calculations. Depending on the degree of rain coverage, an orbit of PR data can require anywhere from several hours to several days of wall clock time to be produced. Due to the *ad hoc* nature of the synthetic data production process, there are no complete libraries of data available on disk, although there are several sample data sets for each of the instruments. Since each algorithm developer has unique needs, data has been generated according to specific requests.

The turn-around times for synthetic data requests have varied depending upon the type of request:

- *"Standard" synthetic data.* For requests that have required no changes to the instrument simulations, the turn-around time has usually been a day or two, with the results being disseminated via ftp sites.
- *"Custom" synthetic data.* For requests that have required modification to the simulations, the turn-around time has ranged from a few weeks to a few months, depending upon the complexity of the changes involved.
- *Data sets beyond the scope of the instrument simulations.* For requests which have not been feasible to completely fulfill, the data developer has worked closely with the algorithm developer to generate data sets which fulfill as many of the scientist's requirements as possible.

5.2 DATA SETS FOR V3 ALGORITHM DELIVERY

TSDIS has created Level 1B synthetic data sets for TMI, PR, and VIRS for use in developing and testing algorithms. Three different models were used to generate the L1 PR data: narrow bin has a small 1 km footprint, toga 240 has relatively more stratiform clouds, and toga 210 has relatively more convective clouds. These data sets are small, only 150 or 250 scans. These data are available through the TSDIS Toolkit Homepage.

TSDIS will also make available, upon request, up to 3 days of Level-1B, Level-1C and Level-2 satellite data to algorithm developers. The Level-2 data will be generated by executing Level-2 algorithms.

<u>6. TSU PRODUCT INGEST</u>

TSDIS will get 3A-46 by anonymous ftp from agnes.gsfc.nasa.gov and verify the reception by e-mail. The algorithm developer will make monthly datasets available.

7. ABBREVIATIONS AND ACRONYMS

The following is a list of abbreviations and acronyms used in this document.

A	
ANSI	American National Standards Institute
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
AVHRR	Advanced Very High Resolution Radiometer
C	
CASE	Computer Aided Software Engineering
CCB	Configuration Change Board
CCR	Configuration Change Request
CDRL	Contract Document Requirements List
COTS	Commercial Off The Shelf
CPU	Central Processing Unit
D	
DID	Data Item Description
E	
EOS	Earth Observing System
EOSDIS	Earth Observing System Data and Information System
F	
FDD	Flight Dynamics Division
G	
GSFC	Goddard Space Flight Center
GSOP	Ground System and Operations Project
GV	Ground Validation
H	
HDF	Hierarchical Data Format
I	
I/O	Input/Output
ICP	Interface Control Plan
ICS	Interface Control Specification
IMSL	International Mathematical and Statistics Library
ITE	Integration and Test Environment

L

LAN	Local Area Network

M

mm	millimeter
MOU	Memorandum of Understanding
MSFC	Marshall Space Flight Center

N

NASA	National Aeronautics and Space Administration

P

PGS	Product Generation System
PR	Precipitation Radar

Q

QC	Quality Control

R

RIS	Raster Image Set
R1	Release 1
R2	Release 2
R3	Release 3
RST	Remote Science Terminal

S

SCF	Science Computing Facility
SDS	Scientific Data Set
SGI	Silicon Graphics Incorporated
SSM/I	Special Sensor Microwave/Imager
STD	Standard

T

TBD	To Be Determined
TMI	TRMM Microwave Imager
TRMM	Tropical Rainfall Measuring Mission
TSDIS	TRMM Science Data and Information System
TSU	TSDIS Science User

U

UARS	Upper Atmospheric Research Satellite

V

V3	Version 3
V4	Version 4
VIRS	Visible and Infrared Sensor

8. GLOSSARY

The following is a glossary of terms used in this document.

CASEVision	SGI CASE tool for code debugging and analysis
Toolkit	Collection of functions that facilitate the seamless integration of science algorithm software into the TSDIS environment.
Processing Parameters	Parameters used in algorithm execution; these are passed to the algorithm on the command line.
Synthetic Data Sets	Data generated from simulations and formatted in the TSDIS product file formats.
Operating System	The computer’s control system which manages hardware resources and controls program execution.
Software Library	A library of subroutines available for use by an algorithm.
Production Environment	The environment used to operationally run TRMM algorithms.
Integration and Test Environment	The environment used to test candidate TRMM algorithms.
Submittal Package	Algorithm software and supporting documentation submitted for integration into TSDIS.

APPENDIX A
ALGORITHM SUBMITTAL PACKAGE CONTENTS

A.1. SUBMITTAL PACKAGE

The Science Algorithm Developers will submit their algorithm software along with supporting documentation in the form of an Algorithm Submittal Package when the algorithm is to be integrated into TSDIS. The purpose of the submittal package is to identify the necessary files for properly testing the algorithm in the TSDIS environment and to provide evaluation criteria for sizing the operational TSDIS (production environment) to accommodate the algorithm once it is integrated into TSDIS. The algorithm submittal package for Version 3 and post-Launch algorithms will consist of the following:

a. source code for the algorithm
b. a list of software libraries needed to execute the algorithm
c. make files used to execute the algorithm
d. processing requirements
e. processing parameters to be used with the algorithm (see section A.1.3)
f. the browse generation specifications for the product of the algorithm
g. browse metadata definitions
h. the test data used by the algorithm developer to test the algorithm on the home computing environment (identified or supplied)
i. the results file from the testing efforts on the home computing environment (these constitute the expected results for TSDIS testing)
j. supporting documentation for the algorithm.

A.1.1 Source Code and Software Libraries

The algorithm source code consists of the algorithm software, any files of parameters used in processing, any INCLUDE files required by the algorithm software which are not standard C or FORTRAN INCLUDE files and the make files. It should also include an indication of utilities needed by the software which are not included in the TSDIS toolkit and any software libraries which are needed to execute the algorithm (such as IMSL).

A.1.2 Processing Requirements

The algorithm package submitted by the algorithm developers will include an assessment of the processing requirements of the science algorithm. These data processing requirements include wall clock time and disk space requirements. These requirements reflect the processing performed on the algorithm developer’s home computing system. These will be updated based on algorithm testing in the TSDIS test environment. Additional processing requirements, such as Central Processing Unit (CPU) time and I/O time will be added as a result of the testing on the

TSDIS test environment. These processing requirements constitute the benchmark for the algorithm.

These estimates will be included in the User's Guide (see section A.3) which will be provided to TSDIS with each algorithm. These estimates will be updated to reflect processing parameters resulting from the formal algorithm testing performed on the ITE.

A.1.3 Processing Parameters

It is anticipated that most TSU algorithms will require input parameters for use within the algorithm code. It is expected that, for Level 1 and Level 2 algorithms, these parameters represent the algorithm's required input file(s), output file(s), and any intermediate file(s), if necessary. Static files such as a land/sea mask or a look up table are not passed in via the command line. They are considered part of the algorithm. The algorithm can expect to find the static files in the same directory as the algorithm executable. Since final Level 3 products are generated on a monthly or pentad basis, Level 3 algorithms require additional processing parameters. The list below shows the set and order of processing parameters for level 1B and higher based on the algorithm category. All processing parameters are passed to algorithms as command line arguments, in the form of strings, upon invocation of the algorithm executable. The algorithm executable may be a program or a script, depending on the algorithm.

Satellite - Level 1B through Level 2B
algorithm_executable *<input file id(s)> <output file id> <intermediate file id(s)>*

GV - Level 2A (excluding 2A-52)
algorithm_executable *<input file id(s)> <output file id> <intermediate file id(s)>*

GV - Level 3~~,~~ and 2A-52
algorithm_executable *<input file id(s)> <output file id> <intermediate file id(s)> <begin/mid/end of period flag> <year> <pentad/month>*

Satellite - Level 3
algorithm_executable *<input file id(s)> <output file id> <CSI file id> <intermediate file id(s)> <begin/mid/end of period flag> <year> <pentad/month>*

where:

input file id(s) -	input data files required by the algorithm. These are typically science product files created by another algorithm or data product from outside TSDIS.
output file id -	single primary output of the given algorithm.

CSI file id -	output coincidence file
intermediate file id(s) -	intermediate file id(s) read or written by the current execution of the algorithm.
begin/mid/end of period flag -	a flag which indicates whether the current invocation of the algorithm executable is for the beginning, middle, or or end of the period (pentad/month). The values are "BEGIN", "MIDDLE", or "END". If this flag indicates the middle of the period, the algorithm executable computes daily accumulation statistics using its intermediate file(s). If this flag indicates the beginning of the period, the algorithm executable also performs any necessary initialization. If this flag indicates the end of the period, the algorithm executable also performs steps to finalize the product. This flag is a character string.
year -	4-digit year in which the current input data were collected, e.g. “1998”. The year is written as a character string.
pentad/month -	the number of the current pentad or month being processed. This number is written as a character string.

The list below shows the set and order of processing parameters for each algorithm (Level 1B and higher) based on our current understanding. For clarity, the algorithm executable name (program or script name) appears in the list below as L<algorithm>main:

L1B01main *<L1A VIRS file id> <ephemeris file id> <L1B01 output file id>*
<L1B01 intermediate instrument analysis data file id>
<L1B01 intermediate instrument analysis report file id>
<L1B01 intermediate instrument limit checks file id>
<L1B01 intermediate geolocation limit check file id>

L1B11main *<L1A TMI file id> <ephemeris file id> <L1B11 output file id>*
<L1B11 intermediate instrument analysis data file id>
<L1B11 intermediate instrument analysis report file id>
<L1B11 intermediate instrument limit checks file id>
<L1B11 intermediate geolocation limit check file id>

L1B21main *<L1A21 file id> <ephemeris file id><L1B21 output file id>*

L1C21main *<L1B21 file id> <L1C21 output file id>*

L2A12main *<L1B11 file id> <L2A12 output file id> <L2A12 verification intermediate file id>*

L2A21main	*<L1B21 file id> <L2A21 output file id> <L2A21 intermediate temporal read file id> <L2A21 intermediate temporal write file id> <L2A21 intermediate spatial file id> <L2A21 verification intermediate file id>*
L2A23main	*<L1C21 file id> <L2A23 output file id> <L2A23 intermediate verification file id>*
L2A25main	*<L1C21 file id> <L2A21 file id> <L2A23 file id> <L2A25 output file id> <L2A25 intermediate verification file id>*
L2B31main	*<L1B11 file id> <L1C21 file id> <L2A21 file id> <L2A23 file id> <L2B31 output file id> <L2B31 intermediate verification file id>*
L3A11main	*<L1B11 file id> <L3A11 output file id> <L3A11 intermediate file id> <begin/mid/end of period flag> <year> <month>*
L3A25main	*<L1C21 file id> <L2A21 file id> <L2A23 file id> <L2A25 file id> <L3A25 output file id> <L3A25 intermediate file 1 id> <L3A25 intermediate file 2 id> <L3A25 intermediate file 3 id> <L3A25 intermediate file 4 id> <L3A25 intermediate file 5 id> <L3A25 intermediate verification file id> <begin/mid/end of period flag> <year> <month>*
L3A26main	*<L1C21 file id> <L2A21 file id> <L2A23 file id> <L2A25 file id> <L3A26 output file id> <L3A26 intermediate file id> <L3A26 intermediate verification file id> <begin/mid/end of period flag> <year> <month>*
L3B31main	*<L2A12 file id> <L2B31 file id> <L3B31 output file id> <L3B31 intermediate file id> < L3B31 verification intermediate file id> <begin/mid/end of period flag> <year> <month>*
L3B42main1	*<L1B01 file id> <L2A12 file id> <L3B31 file id> <L3B42 intermediate Geo-IR calibration parameters file id> <L3B42 intermediate VIRS clipped file id> <L3B42 intermediate TMI clipped file id> <L3B42 intermediate TMI unclipped file id> <L3B42m1 intermediate verification file id> <begin/mid/end of period flag> <year> <month>*

L3B42main2 *<L3B42 intermediate Geo-IR calibration parameters file id>*
<L3A44 file id> <L3B42 output file id>
<L3B42m2 intermediate verification file id>
<year> <day of year>

L3B43main <input file>
Where input file is an ASCII file with inputs, one to a line. The inputs are:
<L3B42 intermediate TMI unclipped file id>
<L3B42 intermediate Geo-IR calibration parameters field >
<L3B31 file id>
<L3A46 file id>
<L3A45A or L3A45B file id>
<L3B43 output file id>
<L3B43 intermediate verification file id>
<year>
<month>
<N>
<N 3B42 file ids, one to a line>

L2A52main *<L2A53 file id> <L2A52 output file id>*
<begin/mid/end of period flag> <year> <month>

L2A53main *<L2A53 switch>*
<L1C51 file id> <L2A54 file id> <L2A53 output file id>

L2A54main *<L1C51 file id> <L2A54 output file id>*

L2A55main *<L1C51 file id> <L2A54 file id> <L2A55 output file id>*

L3A53main *<L2A53 file id> <L3A53 output file id> <L3A53 intermediate verification file id>*
<L3A53 intermediate accumulation file id>
<L3A53 intermediate administration file id>
<begin/mid/end of period flag> <year> <pentad>

L3A54main *<L2A53 file id> <L3A54 output file id> <L3A54 intermediate verification file id>*
<L3A54 intermediate accumulation file id>
<L3A54 intermediate administration file id>
<begin/mid/end of period flag> <year> <month>

L3A55main *<L2A55 file id> <L3A55 output file id> <L3A55 intermediate verification file id>*
<L3A55 intermediate CFAD file id>
<L3A55 intermediate vertical profile file id>
<begin/mid/end of period flag> <year> <month>

A.1.4 Browse Specification

A.1.4.1 Product Browse Definition

Browse products for TSDIS are intended to display a subset (sub-sampled data set or summary tables) of TRMM data products, to be used for inspection and ordering. Typically there will be only one browse product for each algorithm. It is the Algorithm Developer's responsibility to state explicitly what the browse product will be for each algorithm. Where multiple variables are anticipated in the browse product (such as PR), the variable to be used for map overlay correlation must be indicated. The Algorithm Developers must also indicate aspects of the browse product requirements such as color look-up table to be used to display the image, dimensions, parameters to use, channels to select, etc. The Browse products file specifications for the at-launch algorithms are defined in Volume 5 of the ICS. Any new algorithms requiring changes or additions to the Browse must indicate this as part of the algorithm submittal package.

A.1.4.2 Browse Metadata

TSUs will use the browse product to assist in selecting data requests. The TSUs will define the search parameters they need for browsing TSDIS data. These will be included in the Browse metadata.

A.1.5 Testing Datasets

The test input data submitted as part of the algorithm submittal package are the data used by the algorithm developers to test the algorithm on their own computing facilities. TSDIS will use these input datasets as part of the initial checkout of the algorithm to verify that TSDIS can produce the same results as the algorithm developer, using the same input data. If the data already reside at TSDIS, then identification of the data to be used is sufficient. Otherwise, the TSU should supply the datasets used as inputs during their algorithm testing on their home computing environment.

The expected output data submitted as part of the algorithm submittal package are the results produced by the algorithm on the home computing facilities. TSDIS will use these expected output datasets as part of the initial checkout of the algorithm to verify that TSDIS can produce the same results.

A.1.6 Supporting Documentation

The supporting documentation consists of a description of the algorithm (what does it produce and how), an algorithm identification (e.g. 2A-21), a description of the tests run on the home computing environment (how the system was set up, which files contain which input test data, test files, any computer environment variables which need to be set up, etc.) and a User's Guide for the software (see section A.3). The use of environmental variables is not recommended.

A.2 PACKAGE DELIVERY FORMAT

The source and test files should adhere to ANSI C and ANSI FORTRAN conventions. Extensions to FORTRAN 77 are listed in Section 4.1 of the ICS Volume 1. The documentation (such as the User's Guide) should be either in Microsoft Word 6.0 or an ASCII text file~~s~~.

A.3 USER'S GUIDE CONTENT

The User's Guide includes such information as the processing requirements of the algorithm (initial estimates from the results of the home computing environment efforts - these will be updated with TSDIS computer environment results once the software is ported to the TSDIS environment), instructions on how to run the software, and what to do when an error occurs. The following presents the outline and content of the Users' Guide submitted with the algorithm for integration into TSDIS. Text which is standard (boiler plate) for all TSDIS Users' Guides is denoted by *italics*. Paragraph title standards are denoted by bold type. The following User's Guide outline is based on the user's guide outline for the Upper Atmospheric Research Satellite (UARS) project.

1.0 Processing Overview
The purpose of the <name of processing software> *is to* <describe the purpose of the software at a very high level such as "convert the level 1B product into Level 2A products as a function of.." Describe what processing steps are required and what each step consists of.>

2.0 Overview of <name of software routine performing first job step>
Describe the overall file and operator manipulation required for performing the job step. Describe how to compile, link, and run the algorithm.

2.1 Required Resources for <name of software routine performing first job step>
Describe the required computer system resources for the software routine to include CPU usage, wall clock time, output memory requirements, and ancillary data memory requirements.

2.2 Input Parameters Descriptions
List and describe the input parameters needed for the execution of the program.

2.3 Input/Output File Descriptions
List and describe the calibration files, auxiliary data files, and ancillary data files needed by the software routine. Also list and describe the scratch data files and product data files generated by the software routine.

2.4 Processing Assumptions and Restrictions
Describe the assumptions about preceding processing (such as it is assumed all processing of level 1A data is completed before processing of this software routine commences) and the restrictions on the current processing (such as data must be processed in time sequence order).

2.5 Libraries for <name of software routine performing the job step>
List the standard UNIX environment library routines needed by the software routine to execute. List any software libraries (and routines) needed to compile and link the software. Also list the non-standard (i.e. programmer created) routines needed by the software routine to execute.

2.6 Error Messages
List the error and warning messages generated by the program and what action should be taken by the TSDIS scheduler or operator.

2.7 Special Conditions
Describe the steps to be performed in the event of a software failure (such as: a restart is necessary and any data files generated should be deleted). Describe any error handling procedures. Describe which explicit exit codes, if any, are used by the algorithm.

[For each software routine performing a new job step, repeat paragraphs 2.0 through 2.7 with numbering starting at 3.0]

APPENDIX B
RECOMMENDED PROGRAMMING PRACTICES FOR ALGORITHM DEVELOPERS

B.1 OVERVIEW

The structure of a software program can greatly influence the computing time required for execution. Additionally, certain programming practices impact the maintainability and testability of the software. The Algorithm Developers are encouraged to employ the following types of programming practices during their algorithm development. Potentially, the Algorithm Developers could use a software metrics tool to suggest restructuring of code to reduce processing time, reduce the number of testing/validation test sets, reduce testing time, and make code more maintainable.

B.2 PROGRAM STRUCTURE

Structured constructs such as the "IF .. THEN .. ELSE", "IF .. THEN.. ELSEIF", and "CASE", when used appropriately, will reduce processing time for an algorithm. Use of these constructs to isolate a specific instance or range of values of a data item(s) will reduce the times the data item is checked for particular values. If a certain value is more dominant than others (or expected to be) then checking for this case first will reduce the amount of data item checking performed for the algorithm. Redundant data value checking (checking the same data item for several different values in a series of statements) wastes CPU time. This is especially true when these comparisons are made for several iterations of the data set (i.e., the check is made within a control loop of some sort). The use of a structured construct (such as "IF .. THEN .. ELSEIF" or "CASE") is a better implementation. The following is an example of this logic (actual instructions will depend on language selected for implementation).

ORIGINAL CODE LOGIC:

```
If X < -0.01 then X=-0.01          |Always Executed
If X>0.5 then X=0.5                |Always Executed
If X=0 then X=0.01                 |Always Executed
```

CHANGED CODE LOGIC:

```
If X<-0.01 then X=-0.01            |Always Executed
        Elseif X> 0.05 then X=0.05     |Executed only when first check fails
                Elseif X=0 then X=0.01 |Executed only when first and second checks fail
```

Also, if there are a series of checks involving multiple data items mixed with a series of checks involving single data items, then the checks for the single data items should be performed first and the other combinations "skipped" if the single data item condition is true. This "skipping" of checks once a condition is found to be true will alleviate the computing time burden of checking for conditions which are known to be untrue or which are unnecessary to check for since a prior condition was already met. Programs involving multiple and large numbers of iterations involving these types of checks will realize significant savings in execution time resulting from minor re-structuring of the implementation of the algorithm.

For example, the following checks could more efficiently be constructed as indicated:

Example 1:
ORIGINAL CODE LOGIC:

```
If (A>10 and B<5) or
   (A>12 and C>4 and D=1) or
   (X>14 and D>2 and Z<3) or
   (Y=10)
then K=1
else K=2
```

CHANGED CODE LOGIC:

```
if Y=10 or
  (A>10 and B<5) or
  (A>12 and C>4 and D=1) or
  (X>14 and D>2 and Z<3)
then K=1
else K=2
```

Of course if Y=10 is a condition which is rare, then placing this condition first is NOT a time saver.

Example 2:
ORIGINAL CODE LOGIC:

```
If (A>10 and B<5) or
   (A>10 and C>4 and D=1) or
   (A>10 and D>2 and Z<3) or
then K=1
else K=2
```

CHANGED CODE LOGIC:

```
if (A>10)
then if ((B<5) or
         (C>4 and D=1) or
         (D>2 and Z<3)
         then K=1
else K=2
```

Performing I/O routines outside of computational loops will reduce the overhead time involved with opening and retrieving data. Also, code which is not part of the loop (e.g. initializations) should be moved to outside of the loop.

B.3 MAINTENANCE PRACTICES

Maintenance of algorithm source code potentially involves changing the source code to correct computations or parameters used by the algorithm. This is the responsibility of the Algorithm Developers. TSDIS will not do any maintenance unless the TSUs approve. Maintenance may also involve making changes to source code and/or script files to accommodate computer environment changes and software utility package changes. Implementing these changes is simplified and less prone to errors when certain programming practices are followed. Additionally, TSDIS personnel working on a system may change over the life of the system. Implementing the following programming practices will ease the transition of new personnel into the TSDIS environment.

Passing functions as arguments to other functions makes it difficult to trace problems in the code. TSUs should refrain from passing functions as arguments to other function calls. Additionally, passing a large number of parameters in a subroutine or function call complicates any modifications to the routine. It is simpler to construct a data grouping (e.g., structure) composed of the necessary data (passed parameters) and then pass the grouping. A good "rule of thumb" is to limit the number of passed parameters to seven.

The understanding of the code is greatly enhanced by the use of structured constructs such as "IF..THEN..ELSE", "DO WHILE", "DO UNTIL", "CASE", and "FOR...LOOP". These constructs are preferred to using the "GO TO" construct because the "GO TO" construct results in code which is more difficult to test and understand.

Commenting of source code will aid TSDIS personnel in the maintenance efforts. A well commented program enhances the ability of developers, testers, and code maintainers to understand what is happening in the code.

Overly complex code is difficult to adequately test and even more difficult to maintain. Code complexity is not necessarily tied to algorithm complexity. Code complexity is more a measure of the degree of structure of the code. Development of comprehensive test cases for programs ensures that all cases are tested prior to integration into TSDIS. Structured coding techniques aid in identifying these test cases and can result in fewer, less complicated test scenarios.

B.4 UNDESIRABLE PROGRAMMING PRACTICES

The following practices should be avoided since they will hinder algorithm integration into the TSDIS and/or cause unnecessary burdens on the processing capabilities of the TSDIS system. These practices will not result in rejection of the code but they are discouraged.

Repeated blocks of the same code should be avoided (including COMMON statements). Creating a subroutine or function for these sequences of steps is a better implementation (depending on the number of times the block of code is repeated). Using repeated blocks of code means that a change in this sequence of steps must be duplicated in several places. This procedure is prone to errors due to incomplete modification.

Use of equivalence statements should be avoided.