

# MODIS SCIENCE DATA SUPPORT TEAM PRESENTATION

March 27, 1992

## AGENDA

	<u>Page</u>
1. Action Items .....	1
2. MODIS Airborne Simulator (MAS) Status .....	2
3. Intergraph Visit .....	4
4. MODIS SDST Deliverables .....	5
5. Guidelines for TMS .....	6

ACTION ITEMS:

01/03/92 [Team]: Check on the set of software engineering tools available in Code 530 to see if any of these would be of use to the SDST. (Arrangements are being made to have Frank McGarry come to one of the MODIS SDST meetings and talk about the tools they use and would recommend. This will have to occur after the science team meeting because of Frank McGarry's schedule.) STATUS: Open. Due date 02/14/92.

01/17/92 [Tom Goff]: Have a polished version (with peer review) of the file dump routine ready for the MODIS Science Team Meeting. (Copies of the finished version together with RDC internal review comments were given to Ed Masuoka to pass on to Will Webster for use by the SDST software review committee.) STATUS: Open. Due date 04/01/92.

02/21/92 [Ed Masuoka]: Talk to Code 930 and find out what tools they have for porting data between computers from different vendors. (Ed says the U-Lab is purchasing QA Fortran and QA C. He will talk to Bill Mish about data conversion.) STATUS: Open. Due date 03/13/92.

02/21/92 [Lloyd Carpenter and Team]: Identify a list of risks associated with porting Team Members' algorithms to the PGS. Prepare these for discussion at the Science Team Meeting. STATUS: Open. Due date 04/01/92.

03/20/92 [Liam Gumley]: Make a list of candidate algorithms to use MAS data (to be discussed at the Science Team meeting). Start with the atmospheres group. (See report in handout.) STATUS: Open. Due date 03/27/92.

03/20/92 [Tom Goff]: Ensure that every problem identified in porting Mike King's code is addressed in the Team Member coding guidelines. (See draft version of guidelines in handout.) STATUS: Open. Due date 03/27/92.

03/20/92 [Lloyd Carpenter]: Gather the MODIS Data Product Attributes information, and write a cover letter to Team Members for updating the information, and discussion at the Team Meeting. (Cover letter written and delivered.) STATUS: Open. Due date 03/27/92.

## MODIS Airborne Simulator status (Liam Gumley)

### Progress up to 26 March 1992

#### (1) MAS data processing status

<u>Flight Date</u>	<u>Area covered during flight</u>	<u>Level-0 data received</u>	<u>Processing completed</u>	<u>INS offset fixed</u>
10/31/91	Ames test flight CA/NV	yes	3/3 tracks	yes
11/12/91	Ferry flight CA to TX	yes (subset)	1/1 tracks	no
11/14/91	Coffeyville KS	yes	16/16 tracks	no
11/18/91	Coffeyville KS	yes	14/14 tracks	yes
11/21/91	Coffeyville KS	yes		
11/22/91	Coffeyville KS	yes		
11/24/91	Gulf coast TX/LA	yes		
11/25/91	Coffeyville KS	yes		
11/26/91	Coffeyville KS	yes		
12/03/91	Gulf coast TX/LA	yes		
12/04/91	Gulf coast TX/LA	yes		
12/05/91	Coffeyville KS	yes	29/29 tracks	no
12/07/91	Coffeyville KS	yes		
11/16/91	Ground visible calibration	yes	10481 scanlines (no navigation)	
11/20/91	Ground visible calibration	yes	6078 scanlines (no navigation)	
11/23/91	Ground visible calibration	yes	10281 scanlines (no navigation)	

---

Processing was delayed to some extent by a problem with the NFS link from the LTP Iris to the VAX. The problem was resolved on Wednesday. While the NFS link was down, processing could only take place on the VAX. This prompted me to move a complete up-to-date set of processing code from the Iris to the VAX and recompile and test all programs. The VAX was used to produce 2 of the 14 flight tracks from 18 November 1991. The output data produced is normal in all respects, even though the VAX is still slow compared to the Iris (around 7 hours processing on the VAX versus 1 hour on the Iris).

#### (2) Candidate algorithms from the MODIS Atmospheres Group for use with MAS data

I spoke to several members of the MODIS Atmospheres Group and asked them what algorithms they were using, or were planning to use with the MAS data.

Mike King reported that his main processing algorithm was contained in the Cloud Optical Depth (CLDOPT) code. The code is currently executing on a CRAY UNICOS system, and was being updated/modified/beautified in order to work with MAS data. Si Chee Tsay is working on this code, and it is expected that a version suitable for porting will be ready sometime in May. It was also suggested that noise filtering/removal would form a part of future algorithm development.

Paul Menzel was unavailable for discussion before 03/27/92. I spoke to Chris Moeller who works with Paul at Wisconsin. There are several people working on algorithm and code development for MAS data. They do have code running currently for several tasks, but most of the code is still under development. A critical aspect is that the MAS will have appropriate spectral channels for some algorithms for the first time in the ASTEX deployment (June '92).

Although algorithms and code are still under development, Chris did not think there would be a problem in providing a version for our prototyping purposes.

Yoram Kaufman and Didier Tanré are chiefly responsible for aerosol algorithms for MODIS. Yoram does intend to eventually use data from MAS for aerosol, water vapor, and fire detection algorithm development. He expects the first MAS data applicable to this task will be from the Brazil biomass burning field experiment in August 1993. There are several possible algorithms available for estimation of these parameters, however no code exists at the moment that could be applied to MAS data directly. However it should be noted that his algorithms will directly use algorithms from radiative transfer codes such as LOWTRAN7 and 5S/6S. These codes perform the bulk of the 'science' algorithm computation, and the remaining code would perform functions like datafile manipulation, equation solution/inversion etc.

The following is a list of parameters to be derived from MODIS (King et. al., 1992) by the Atmosphere Group, and which could also be used for MAS data.

Parameter	Responsible Investigator
-----------	--------------------------

*Cloud properties*

Optical Thickness	King
Effective radius	King, Menzel
Thermodynamic phase	King
Cloud top pressure	Menzel
Cloud top temperature	Menzel
Effective emissivity	Menzel
Cloud fraction	King

*Aerosol properties (land)*

Optical thickness	Kaufman, Tanré
Effective radius	Kaufman, Tanré
Single scattering albedo	Kaufman, Tanré
Mass loading	Kaufman, Tanré

*Aerosol properties (water)*

Optical thickness	Kaufman, Tanré
Size distribution	Kaufman, Tanré
Mass loading	Kaufman, Tanré

*Water vapor properties*

Precipitable water vapor (land)	Kaufman, Tanré
Precipitable water vapor (water)	Menzel
Atmospheric stability	Menzel

All of these parameters are retrievable from MAS, given the appropriate spectral channels. Some of them require ancillary data such as temperature profiles, radiative transfer computation results etc.

Reference: M.D. King, Y.J. Kaufman, W.P. Menzel, D. Tanré, "Remote Sensing of Cloud, Aerosol, and Water Vapor Properties from the Moderate Resolution Imaging Spectrometer (MODIS)", *IEEE Trans. Geosci. Remote Sensing*, vol. 30, pp 2-27, 1992.

## Visit to Intergraph Corporation

25 March 1992

Representatives from the MODIS SDST visited the Intergraph Federal Systems Division in Virginia to obtain a better understanding of the capabilities of the Intergraph work stations as might be applied to the registration and rectification tasks for ground locating the MODIS instrument data. We were treated to three presentations consisting of an Intergraph corporate overview, a demonstration of their display work station, and a slide show of intellectual capabilities. The demo allow several questions to be asked and the presentations were followed by an informal discussion of the MODIS task.

The demo was performed on one of their 27", 1660x1280 32bit color systems. These systems use the Intergraph Clipper computer architecture (ex Fairchild) with Vitek graphics boards. This system started as a full featured GIS system to which remote imaging has been added. The software remote sensing capabilities are derived primarily from the use of commercial Landsat and Spot imagery. These images are used to correct GIS data and to provide type classification as an adjunct to the normal GIS functions. Their system incorporates a very capable interactive ground registration technique utilizing statistical indications for goodness of fit and several resampling techniques to warp the image data either to other images or to GIS vectors. We did not see any indication of the use of mapping projections or satellite ephemeris capabilities, although the glossy brochure mentions a mapping projection interchange package.

We discussed the MODIS problems of having a variable IFOV coverage and a variable IFOV overlap to the resident personnel. They volunteered that there were company employees in Huntsville that are familiar with these concerns and gave us a name to contact for further information. They also provided us with a contact with a company in St. Louis that has had considerable experience in correcting satellite ephemeris data from remotely sensed ground scenes by incorporating DEM models and back solving.

This system reportedly supports open system standards but they were not currently running X-windows or other open systems software. They have published their database standards and details to allow user to write interfaces to this database. They also mentioned that users could write their own modules that can be interfaced into the Intergraph system. The software used in the demo (MicroSystem?) has been partially ported to PC's, Macintoshes, and other UNIX operating systems.

Although this system with it's large monitor and true 3-D graphics would be a nice present from Santa Claus, it currently does not have a sufficiently complete suite of software tools, for our use as MODIS investigators, to justify it's \$100,000 price tag.

# DRAFT

## MODIS SDST Deliverables

- 1 MODIS SDST Schedule
- 2 MODIS Software and Data Management Plan
- 3 MODIS Team Leader Computing Facility Plan
- 4 Guidelines for MODIS Team Members Science Algorithms
- 5 MODIS Airborne Simulator (MAS)
  - 5.1 Version 1 Software
  - 5.2 Data Users Guide
  - 5.3 Data Catalog
  - 5.4 Version 1 Metadata
  - 5.5 Level-1 Processed Data
  - 5.6 Prototype Level-2 Algorithm List
- 6 MODIS Level-1 Software Development Plan
- 7 MODIS Level-1A Design
- 8 MODIS Level-1B Design
- 9 MODIS Control Shell Development plan
- 10 MODIS SDST Training Plan
- 11 MODIS SDST Project Plan

**DRAFT**

**MODIS Science Data Support Team (SDST)  
Guidelines for MODIS Team Members Science Algorithms**

# DRAFT

## Table of Contents

1	Introduction . . . . .	1
	1.1 Objective . . . . .	1
	1.2 Languages/Operating System . . . . .	1
	1.3 Programming Style . . . . .	1
2	Readability/Documentation . . . . .	2
	2.1 Documentation . . . . .	2
	2.2 Declarations . . . . .	3
	2.3 Variable Names . . . . .	3
	2.4 Structure . . . . .	3
	2.5 Modularity/Cohesiveness . . . . .	3
3	Portability . . . . .	4
	3.1 Language/Operating System . . . . .	4
	3.2 Binary Data . . . . .	4
4	Testability . . . . .	4
	4.1 Module Testing . . . . .	4
	4.2 Test Drivers . . . . .	5
5	Maintainability . . . . .	5
	<u>Appendix A: Software Code Evaluation Criteria</u> . . . . .	A-1
	References: . . . . .	R-1

# DRAFT

## 1 Introduction

This is the first draft version of the guidelines for MODIS Team Members science algorithms. The current schedule calls for the next update in January 1993. The  $\beta$  version is scheduled for July 1993, Version 1 and Version 2 are scheduled for July of 1994 and 1995 respectively. Corrections and suggested changes are welcome at any time.

### 1.1 Objective

The objective of these guidelines is to facilitate the porting, integration, testing, documentation, and maintenance of algorithms for the generation of MODIS science data products on an operational basis. The MODIS Science Data Support Team (SDST) will generate a MODIS Software and Data Management Plan which will address the development, validation, integration, operational testing, documentation, maintenance, modification, and configuration management of the MODIS science data processing algorithms. These guidelines are designed to assist the MODIS Science Team Members in preparing their algorithms for this process.

### 1.2 Languages/Operating System

All MODIS science data processing algorithms to be ported to the MODIS Team Leader Computing Facility (TLCF) and integrated into the control shell with other algorithms will be written in either Standard Fortran or Standard C programming language, without extensions. Any exceptions should be brought to the attention of the MODIS Team Leader and coordinated with the SDST. The EOS Product Generation System (PGS) and the MODIS TLCF are planned to conform to the Portable Operating System Interface for UNIX (POSIX) and the Government Open Systems Interconnections Profile (GOSIP).

### 1.3 Programming Style

The primary goal of these guidelines is to facilitate the porting, integration, testing, documentation, and maintenance of algorithms for the generation of MODIS science data products on an operational basis. These guidelines must be consistent with

## DRAFT

accepted good coding practices, but they are not intended to go beyond the present purpose. Choice of programming style, computational methods, etc. is left to the Science Team Members.

## 2 Readability/Documentation

Efficient porting, integration, testing, documentation, and maintenance of algorithms depends heavily on being able to read and understand the code (including the documentation). The code must be written and documented so as to be read, understood, and used by another knowledgeable person or group of people in a different environment. Readability, or understandability, is the most important criterion for successful implementation of MODIS algorithms.

### 2.1 Documentation

Proper software documentation covers a wide range of topics including theory, programmer's guide, user's guide, etc. While all available algorithm documentation should be provided to the SDST, the present focus is on internal documentation of source code.

Comments are interspersed throughout a module, whereas documentation occurs at the beginning of a module. Documentation is specific, and it should be revised to reflect changes. Completeness and readability, not brevity, are the main considerations. For each algorithm module the documentation should, at a minimum, consist of:

- a standard prologue,
- purpose of the module
- definition, description and units of the input, output and internal variables
- method used, if applicable
- references to the literature
- notes and warnings (conscious design limitations).

Input variables should be given thorough and complete

## **DRAFT**

definitions and descriptions. Units, type, dimensions, special cases, upper and lower limits, usage examples, default values, and interrelations with other input variables should all be provided.

### **2.2 Declarations**

Declarations and data statements should be placed just before the first executable statement of the module.

### **2.3 Variable Names**

All variable names should be meaningful to the knowledgeable reader. Avoid the use of short, cryptic variable names. Also avoid the use of language keywords in variable names.

### **2.4 Structure**

The code should be adequately and logically blocked, commented and indented to clearly show the structure and logical flow. Comments should be uniformly set-off from the code. All transfers of control and destinations should be clearly annotated. A path must be defined for every possible outcome of a logical decision. The level of nesting should be kept to a minimum. Statement labels (if used) should occur in a clear and natural sequence. There should be only one statement per line.

### **2.5 Modularity/Cohesiveness**

Complex algorithms can generally be separated in a logical way into modules (subprograms) each of which does a single task. When each module is cohesive, and the coupling between modules is loose, the algorithm becomes more understandable, more testable, more maintainable, and more easily documented. The aim is to achieve a level of modularity which keeps the individual modules cohesive and comprehensible while avoiding the clutter of too many relatively trivial modules. Avoid combining several functions together arbitrarily. Modules should generally be limited in length to 1 or 2 pages.

# DRAFT

## 3 Portability

Portability is of primary concern in the case of MODIS science algorithms which are developed in one environment and implemented for production processing in a different environment. The degree of concern increases with the degree of difference between the two environments.

### 3.1 Language/Operating System

The best guideline to achieve portability is to adhere to standard FORTRAN or C, and avoid the use of system-dependent features.

Many portability problems associated with code developed on a non-UNIX operating system are best avoided by specifying a small set of primitive operations for accessing the environment. Operating system dependencies are then confined to a small number of procedures and functions, so the algorithm can be moved to a UNIX system where the primitives can be implemented.

### 3.2 Binary Data

Special care must be taken to avoid problems associated with porting binary data. Mixing floating point numbers with integers in a single array is especially troublesome, especially when porting to a machine which does not support the same bit lengths. Another concern is the change in roundoff arising from differences in the bit-length of floating point arithmetic in different computers.

## 4 Testability

All test data and test results must be included with the delivery of the algorithm and documentation to the SDST. The porting and integration process is not complete until the algorithm has been successfully tested on the TLCF. These tests must give results which are consistent with testing done prior to delivery to the SDST.

### 4.1 Module Testing

# DRAFT

Each module should be tested independently of the total algorithm. The test should include exercising every logical branch in each module, checking for possible failure, checking for reasonableness of results, comparison of limiting cases with analytic solutions, comparison with published results where possible. Tests should also include cases of invalid or implausible input variables, no matter how unlikely it is that the module will be used incorrectly. In the code, each error message should be kept together with its associated error check.

## 4.2 Test Drivers

Comprehensive test drivers explore all of the branches of an algorithm and compare results with "correct answers" which are included in the driver code. A good test driver will also "failure-test" the model by pushing it into regimes where trouble is expected. Good test drivers, developed with good coding practices, are very helpful in porting, integrating, and maintaining algorithms. Test drivers are not required for MODIS science algorithms, but they are recommended, and they should be supplied to the SDST when available.

## 5 Maintainability

Considering the duration of the MODIS mission, all algorithms will be subject to maintenance (changes, and updating). The maintenance process will be greatly simplified if the guidelines addressed elsewhere in this document are applied in the design and development of algorithms.

# DRAFT

## Appendix A: Software Code Evaluation Criteria

This list of software code evaluation criteria is provided as a suggested checklist for code review. Some of the items apply to more than one category.

### Readability/Documentation

Does each module have the standard prologue?

Are definitions, descriptions and units given for all variables?

Are declarations and data statements placed after the prologue and before the first executable statement?

Are variable names meaningful?

Do variable names avoid the use of language keywords?

Has structured programming been utilized?

Is the code logically and adequately commented, blocked and indented to show structure?

Are comments uniformly set-off from code?

Are all transfers of control and destinations annotated?

Is a path defined for every possible outcome of a logical decision?

Is the level of nesting kept to a minimum?

If statement labels are used, do they occur in a clear and natural sequence?

Is there only one statement per line?

Is the program divided into cohesive and comprehensible modules of reasonable size?

# DRAFT

## Portability

Is the software written in Standard FORTRAN or Standard C without extensions?

Are operating system dependencies limited to a small set of primitive operations?

Are mixed binary arrays of fixed and floating point numbers avoided?

## Testability

Are test data and test results provided with the algorithm?

Are limit checks performed to ensure that variable contents are within the expected range of values?

Are input defaults explicitly tested?

Are errors and associated error messages kept together?

## Other Criteria

### Calling subroutines

Do arguments in call statements not contain arithmetic or logical expressions?

Does each module contain a single entry point and a single exit point?

Are shared variables communicated as arguments whenever practical to ensure program modularity? (Minimize the use of COMMON and EQUIVALENCE.)

### Error handling and prevention

Are flagged/missing data values correctly excluded from calculations?

Is the code designed to handle errors and failures gracefully?

## DRAFT

Are possibilities for infinite loops avoided?

Are stack overflow problems avoided?

### Variables, constants & parameters

Are constants defined? Are counters, variables and parameters initialized?

Are local variables within modules declared as static (type) where appropriate?

Are loop index parameters and array subscripts expressed only as integer constants or integer variables?

# DRAFT

## References:

Warren J. Wiscombe, "Principles of Numerical Modeling with Examples from Atmospheric Radiation" 1989, unpublished.

Upper Atmosphere Research Satellite Software Assurance Recommendations Document, Version 3.0, NASA/GSFC, July 1989