

NAVAL POSTGRADUATE SCHOOL

Monterey, California



19980727 148

THESIS

**ENMOC
MODEL VERIFICATION SYSTEM**

by

Kyongsuk P. Pace

June 1998

Thesis Advisor:

Tim Shimeall

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE June 1998		3. REPORT TYPE AND DATES COVERED Master's Thesis
4. TITLE AND SUBTITLE FNMOC MODEL VERIFICATION SYSTEM			5. FUNDING NUMBERS	
6. AUTHOR(S) Kyongsuk P. Pace				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) Fleet Numerical Meteorology and Oceanography Center (FNMOC) forecasts the atmospheric environment and weather using several meteorological and oceanographic models. These models' forecasting abilities are verified by comparing the model forecast against the observational data and model's analysis. Currently, some models are verified by several inconsistent, maintenance-intensive, non-standardized, and hard-to-use model verification systems designed for a particular model. Some models are not verified because there is no model verification system. This thesis demonstrates the concept of a single model verification system for all FNMOC models to eliminate the inconsistencies and redundancies. The single model verification system standardizes the model verifications and provides the ability to verify those models which are currently unverified. The prototype used a GUI and web browsers to display the model verification statistics. The prototype demonstrates that convenient access to the model verification statistics could aid FNMOC users in evaluating the forecast models' performance. This thesis identifies and documents the user specified verification requirements for several models and implements the most immediate requirements. A complete quantitative model verification system for all FNMOC models will be implemented incrementally, as all the requirements are identified.				
14. SUBJECT TERMS model verification, software engineering, prototype			15. NUMBER OF PAGES 206	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

Approved for public release; distribution is unlimited.

**FNMOCC
MODEL VERIFICATION SYSTEM**

Kyongsuk P. Pace
B.S., Columbus College, 1985

Submitted in partial fulfillment
of the requirements for the degree of

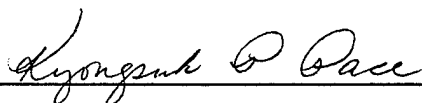
MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

June 1998

Author:

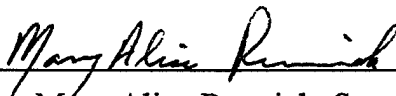


Kyongsuk P. Pace

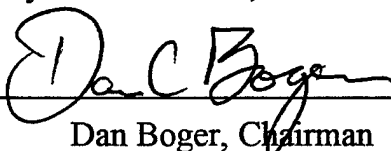
Approved by:



Tim Shimeall, Thesis Advisor



Mary Alice Rennick, Second Reader



Dan Boger, Chairman

Department of Computer Science

ABSTRACT

Fleet Numerical Meteorology and Oceanography Center (FNMOC) forecasts the atmospheric environment and weather using several meteorological and oceanographic models. These models' forecasting abilities are verified by comparing the model forecast against the observational data and model's analysis. Currently, some models are verified by several inconsistent, maintenance-intense, non-standardized, and hard-to-use model verification systems designed for a particular model. Some models are not verified because there is no model verification system.

This thesis demonstrates the concept of a single model verification system for all FNMOC models to eliminate the inconsistencies and redundancies. The single model verification system standardizes the model verifications and provides the ability to verify those models which are currently unverified. The prototype used a GUI and web browsers to display the model verification statistics. The prototype demonstrates that convenient access to the model verification statistics could aid FNMOC users in evaluating the forecast models' performance.

This thesis identifies and documents the user specified verification requirements for several models and implements the most immediate requirements. A complete quantitative model verification system for all FNMOC models will be implemented incrementally, as all the requirements are identified.

TABLE OF CONTENTS

I. INTRODUCTION	1
A. CURRENT MODEL VERIFICATION AT FNMOC	3
B. FNMOC MODEL VERIFICATION SYSTEM	5
C. RESEARCH QUESTIONS	5
D. THE OVERVIEW OF THESIS	6
II. FNMOC MODEL VERIFICATION SYSTEM	9
A. OPERATIONAL RUN	10
B. STATISTICS LIBRARY	11
C. STATISTICS DATABASE	12
D. GRAPHICS	13
E. USER INTERFACE	13
F. MEETING THE REQUIREMENTS	14
III. REQUIREMENTS	17
A. GENERAL REQUIREMENTS	17
B. FNMOC USER REQUIREMENTS	17
C. ASSUMPTIONS	22
D. LIMITATIONS OF RESEARCH	23

IV. DESIGN	25
A. OVERALL STRUCTURAL DESIGN	26
B. DESIGN OF OPERATIONAL RUN (OPS RUN)	27
1. Design of mverif.job (Cray)	30
2. Design of statupd.ksh (Sun)	31
3. Design of Fortran Include File, V_DATA.H	32
4. Design of VEROBS.F90, The Main Verification Against Obs Program	32
C. DESIGN OF STAT LIB	37
1. Map Factors	37
2. Bias (Mean Error)	39
3. Standard Deviations (stdev)	39
4. Root Mean Square (rms or rmse)	40
D. DESIGN OF STATISTICS DATABASE	40
E. DESIGN OF GRAPHICS	42
F. DESIGN OF USER INTERFACE	44
V. EVALUATION	49
A. ALPHA TESTING	49
B. BETA TESTING	51
C. MODIFICATION	52
D. SUMMARY	53

VI. RECOMMENDATIONS AND FUTURE RESEARCH	55
A. RECOMMENDATIONS	55
B. FUTURE RESEARCH	57
1. Ops Run	57
2. Statistics Library	58
3. Statistics Database	58
4. Graphics	58
5. User Interface	59
6. Other Open Issues	60
LIST OF REFERENCES	61
APPENDIX A	63
A. SMS VARIABLES	63
1. NOGAPS	63
2. NORAPS_ASIA	63
3. NORAPS_CONUS	63
4. NORAPS_EUROPE	64
5. NORAPS_IND_OCN	64
6. NOGAPS FOR ASIA	64
7. NOGAPS FOR CONUS	64

8.	NOGAPS FOR EUROPE	64
9.	NOGAPS FOR EUROPE_NEST2	65
10.	NOGAPS FOR EUROPE_NEST3	65
11.	COAMPS_EUROPE	65
12.	COAMPS_EUROPE FOR NEST3	65
13.	COAMPS_SOUTHWEST_ASIA FOR NEST2	66
14.	COAMPS_SOUTHWEST_ASIA FOR NEST3	66
15.	WAM_GLOBAL	66
B.	NAMelist REQUIREMENTS OBTAINED FROM THE USERS FOR EACH MODEL	67
C.	NON-SURFACE NAMelist FILES	73
1.	NOGAPS, all NORAPS and all COAMPS	73
D.	SURFACE NAMelist FILES	73
1.	NOGAPS, all NORAPS and all COAMPS	73
2.	WAM_GLOBAL	74
APPENDIX B SOURCE CODE		75
A.	OPS RUN	75
1.	mverif.job	75
2.	statupd.ksh	81
3.	v_data.h	82
4.	verobs.f90	83

5.	boundary.f90	99
6.	f2ob.f90	103
7.	intgeom.f90	107
8.	uv2df.f90	113
9.	lltread.f90	115
10.	alty.f90	120
11.	raob_qc.f90	125
12.	sfcland.f90	133
13.	sfcship.f90	140
14.	ssmetqc.f90	146
B.	STAT LIB	153
1.	Compute_bias module	153
2.	Compute_rms	155
3.	Compute_std	158
C.	GRAPHICS	161
1.	Plot_data.pro	161
D.	USER INTERFACE	166
1.	Index.html	166
2.	Nogaps.html	166
3.	Noraps_asia.html	168
4.	Noraps_conus.html	170
5.	Noraps_europe.html	171

6.	Noraps_ind_ocn.html	173
7.	Coamps_europe.html	174
8.	Coamps_southwest_asia.html	176
9.	Wam_Global.html	177
10.	Procform.pl	179
11.	Get_data.ksh	181
APPENDIX C		183
A.	STATTEST.F90	183
B.	STATISTICS BY STATISTICS LIBRARY SUBROUTINES	184
C.	STATISTICS BY MICROSOFT EXCEL	185
INITIAL DISTRIBUTION LIST		187

LIST OF FIGURES

1. Context Level Diagram	9
2. Level 1 Diagram	26
3. Ops Run Diagram	28
4. SMS Variables and Description	30
5. Verobs Design	32
6. Namelist Variables and Description	34
7. SQL used to create the generic table	40
8. Statistics Table Attributes	42
9. Graphics Design	42
10. User Interface Design	44
11. Home Page	45
12. WAM_GLOBAL Statistics Page	47
13. WAM_GLOBAL Statistics Graph	48

LIST OF TABLES

1. Inventory of FNMOC statistics routines	12
2. List of user requirements for each models	19
3. Namelist information obtained from the users for each model	68

ACKNOWLEDGMENT

I want to thank FNMOC for allowing me the opportunity to pursue this degree. I want to thank all my friends at FNMOC who encouraged me when I was full of doubts and skepticism. I also want to thank Dr. Mike Zyda for making time in his busy schedule to provide me insightful and valuable advice. My thanks to Dr. Tim Shimeall and Dr. Mary Alice Rennick for their help and guidance on this thesis. Finally, my love and thanks to my husband Duane, my sons Robert, Steve, and Michael.

IDL is a registered trademark of Research Systems, Inc., registered in the United States Patent and Trademark Office, for the computer program described herein. Empress, Empress 4GL and Empress Report Writer are trademarks of Empress Software Incorporated. Netscape and Netscape Navigator are registered trademarks of Netscape Communications Corporation in the United States and other countries.

I. INTRODUCTION

Fleet Numerical Meteorology and Oceanography Center (FNMOC) is a U.S. Navy organization responsible for preparing and disseminating a wide variety of weather and other environmental forecasts. These forecasts support a large and diverse group of civil and military users who are located throughout the world. These forecasts are generated by meteorological and oceanographic models maintained by FNMOC. The accuracy and timeliness of the model forecasts are critical because the model forecasts support operational missions. As part of the quality control process, the model forecasts are periodically verified. While the verification process and frequency vary by model, generally it is an ongoing process. The model verification process is currently accomplished by diverse processes. Generally, all verification systems calculate a variety of summary statistics. These summary statistics are then examined to identify strengths and weaknesses. The models are then modified to improve their forecast performance. These summary statistics are also included in the weekly, monthly, quarterly and annual model performance reports. These reports are read by many meteorological and oceanographic organizations.

This research attempts to provide an enhanced model verification system that streamlines and standardizes the model verification processes in FNMOC. An enhanced model verification system that can be used to verify all FNMOC models would streamline the model verification process. Additionally, a verification system which is easily modifiable to meet new requirements would provide enhanced capability. Furthermore, a system that is easily accessible to the model developers and managements would reduce difficulties in

rapidly assessing verification data.

First, FNMOC needs a common verification system to verify all the model forecasts. Ideally this verification system should enable the users to make comparisons between several different combinations of verification parameters by controlling the independent variables. In other words, a verification system should provide the equivalent comparison statistics for the variable of interest. The comparison might be between models for the same geometry, between the atmospheric levels, between the forecasting period, etc. For example, if a user wants to compare one model's performance with another model's performance, the verification system should provide the equivalent comparison statistics for the different models in the same geometry, same atmospheric level, same forecast periods, etc., thus controlling these commonalities and providing only the true comparison statistics of the model performances.

Second, FNMOC needs a flexible verification system that can accommodate requirement changes. These changes could be an addition of a new model, change in the geometry where the models forecast, addition of new environmental parameters, etc. The requirement changes need to be incorporated immediately. Additionally, these changes should not require extensive modifications to the applications source code when each change is required.

Third, FNMOC needs an easy-to-use tool to access the verification statistics. It should be easy for the model developers and managers to use to generate the needed information in graphical format.

A. CURRENT MODEL VERIFICATION AT FNMOC

Currently, there is no single verification system for FNMOC model developers to use to access and easily analyze the models' performance. There are several model verification systems for different models developed by individual model developers and model verification personnel without adhering to any standards. For example, FNMOC's global meteorological model, Navy Operational Global Atmospheric Prediction System (NOGAPS), is verified by two verification systems, nogstat and verobs. Nogstat verifies NOGAPS forecast against its analysis and verobs verifies against the observational data. FNMOC's regional meteorological models, Navy Operational Regional Atmospheric Prediction System (NORAPS) and Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS), are verified against the observational data by verobs. Verobs compares NOGAPS and NORAPS in the NORAPS regions - Asia, Continental US (Conus), Europe and Indian Ocean. Verobs also provides NORAPS and COAMPS comparisons; however, their geometries do not match exactly. The different geometries add undesirable variability in the model comparison. Another example is FNMOC's global wave model, WAM_GLOBAL. It is verified by a verification scheme developed by the model developer. There is no known documentation and the developer is the only person who has any knowledge about the scheme. There are other systems that can only handle a single model or are very difficult to modify. Additionally, there are redundancies in the model verification efforts among many of the individual verification

systems. Finally, some models do not have a verification system at all.

This lack of standardization also limits FNMOC's ability to rapidly respond to user's new requirements. As an operational center responding to a variety of operational users ranging from the Navy, Air Force and other civil and government organizations, FNMOC frequently receives requests to provide modifications to models. The ability to respond is limited by the fact that the data to compute the model verification statistics are in various formats at various locations. Some data are located in FNMOC's relational database, Integrated Stored Information System (ISIS) on the Cray and the Sun server. The model forecast data (grid data) and the model analysis are stored in the ISIS grid database. The observational data are stored in the ISIS latitude/longitude/time (LLT) database in a format different from the grid data. Some data are stored in a private directory as text files. All of these different data require different reading schemes. The model verification statistics are computed in many different ways since there is no single library to compute the statistics. The model statistics files are in different formats - some in text files, some in the Fortran formatted files, and in different locations - some on the Cray in the operational directory, some in private directories, some on a Sun system.

Finally, the accessibility to and the display of the data in the current systems do not provide the flexibility needed. Accessibility to the verification statistics is very limited in the current verification systems. Only users who have in-depth knowledge about the verification system can access the model verification data. The display of the model verification data is in numerous formats, styles and conventions. Notably, graphical displays with similar purpose are prepared using multiple graphical software packages. This results in a wide

variety of styles and methods used to display similar model statistics. Clearly, there is a strong need for one universal model verification system to verify any FNMOC models and standardize the model verification process.

B. FNMOC MODEL VERIFICATION SYSTEM

The FNMOC Model Verification System is a single, easy-to-modify and easy-to-use model verification system for all FNMOC meteorological and oceanographic models. It was developed by modifying the current FNMOC operating verification systems `verobs` and `nogstat`. The new model verification system standardizes the model verification in FNMOC. It also makes the data and information widely available by leveraging web technology. It currently resides on FNMOC's intranet, but could be placed on FNMOC's internet server to meet the external user's requirements.

C. RESEARCH QUESTIONS

FNMOC model developers need a single, easily modified and easy-to-use model verification system that would standardize the model verification. This research attempted to address these needs through the following questions.

Is it feasible to develop a prototype system that demonstrates the concept of one universal model verification system to verify all FNMOC models? Will this system assist in eliminating redundancies and inconsistencies caused by the present system that uses multiple

model verification methods? Is it feasible to implement a common system for those models that currently do not have a model verification system? Can this standardization of the model verification create a reusable statistics library and consolidate all model statistics in one database in a standardized data format? Can this model standardization be created using graphics with the same format, style and conventions?

Second, can a universal system provide the flexibility needed to rapidly and efficiently modify the model verification system for all FNMOC models? Will this flexibility meet requirements of FNMOC customers?

Third, can a web-based technology provide the access, and ease of use needed to meet the customer and FNMOC personnel needs?

D. THE OVERVIEW OF THESIS

Chapter I of this thesis provides an introduction and problem statement, Chapter II describes the Model Verification System including the background, Chapter III provides the user requirements, Chapter IV provides the design, Chapter V provides the evaluation and Chapter VI provides the recommendations for future research. Appendix A provides the values of the SMS environment variables and namelist values for each model using the system. Appendix B provides the source code. Appendix C provides the prototype testing data.

The figures in this thesis use Yourdon's [Ref. 1] graphical modeling notation. A process is represented by a rectangular box. A process that is further decomposed is represented by a shaded rectangular box. An input or output flow is represented by an arrow.

An external entity is represented by a box with a folded upper left hand corner. A decision point is represented by a diamond. A data store is represented by a drum. A library module is represented as a rectangular box with double side bars.

The `courier` font is used to indicate programs and systems external to the model verification system such as ISIS and HTML. The *italic* is used to indicate the parts of the model verification system such as the *graphics* component.

II. FNMOC MODEL VERIFICATION SYSTEM

The FNMOC Model Verification System is a single model verification system for all FNMOC meteorological and oceanographic models. It replaces the currently existing model verification systems that are redundant, hard to modify and maintenance intensive. It provides an easy-to-use model verification system for the models that do not currently have a verification system. It standardizes the model statistics computations by creating a reusable statistics library. It standardizes the format and storage of the model statistics by consolidating the model statistics in one location using one database management system. It also standardizes the display of the model statistics by using the graphics library routines created with FNMOC's graphics software. This system provides FNMOC model developers a easy-to-use tool to aid them in their analysis of model performance.

It uses web technology to enable the users to make their requests via web browsers and receive the resulting graphics as shown in Figure 1. Implementing the GUI via a web browser eliminates the requirement that the client system have all the necessary software locally, i.e., all the processing is done on the server.

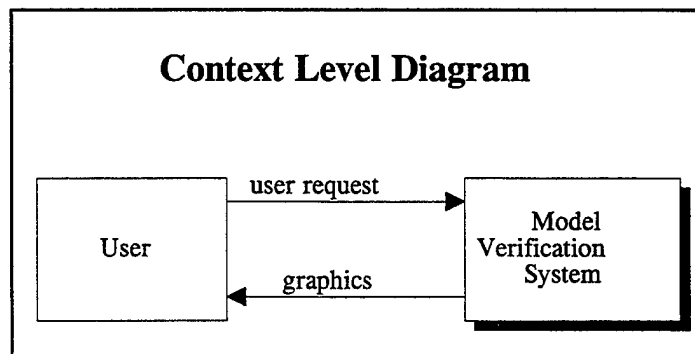


Figure 1. Context Level Diagram

The model verification system has several components. The *operational run (ops run)* component computes the model statistics twice a day after each model run is completed. The *statistics computation library (statistics library)* component is a reusable library that performs the statistics computation for this system as well as other systems (programs) at FNMOC. The *database for the model statistics (statistics database)* component is the single data storage for all the model statistics. The *graphics* component creates the graphs of the model statistics. The *user interface* component receives the user requests, processes them, then returns the resulting graphs back to the user via a web browser. The following sections describe these components in broad, general terms. More specific descriptions of these components are in Chapter IV.

A. OPERATIONAL RUN

FNMOC runs the prediction models twice a day. The model verification statistics are calculated after each operational run of the models. This component has four major parts. First, after the prediction model has successfully completed, the FNMOC operations staff uses the Scheduler Monitor Supervisor (SMS) system to invoke a Korn shell job script, *mverif.job*. This operational job script (*mverif.job*) is run on the same FNMOC Cray computer as the model forecasts, model analysis and the observational data. The *mverif.job* also executes the second and third parts of the operational run components, *verobs* and *veranal*. Additionally, it writes a set of text files that contain the calculated statistics to the appropriate directory and computer. Second, a group of Fortran programs, *verobs*, verifies

the model forecasts against the observational data. Third, a group of Fortran programs, *veranal*, verifies the model forecasts against the model analysis. The model analysis is the initial conditions over a set of grid points covering the forecast region. Fourth, a Korn shell script, *statupd.ksh*, adds the computed model statistics to the model statistics database.

B. STATISTICS LIBRARY

The statistics library has a collection of reusable general purpose statistics programs. These programs are a group of Fortran programs that compute the required statistics for model verification. These include commonly used verification statistics such as mean error (bias), standard deviation of the error (stdev), and root mean squared error (rms). All the programs compute the statistics on the difference between the verifying and predicted sets of data. There is also a program to compute the map factors for different earth projections used by FNMOC.

The current FNMOC routines for computing statistics were modified and adapted in this component. The inventory of the existing statistics computation routines was obtained from the FNMOC Unix Utility Library and the applications under FNMOC configuration management (CM). Table 1 shows the existing Fortran subroutines and the statistics computed by each of them.

Table 1. Inventory of FNMOC statistics routines

Routine	Statistics computed
ancor.f	anomaly correlation (anc) for 10 NOGAPS areas
differp.f, differs.f*	mean bias, mean, sum, standard deviation (std), root mean square (rms)
htrms.f	mean bias, std, rms
nrancor.f	anc
nrhtrms.f	mean bias, std, rms
nrvelrms.f	mean wind u component, mean wind v component, mean wind speed, rms wind speed
stats.f	mean bias, rms, std, threat score, probability of detection (pd), false alarm, skill score
velrms.f	mean wind u component, mean wind v component, mean wind speed, rms wind speed for 10 NOGAPS areas

* The statistics computed do not take any projection into account. All the other subroutines compute the statistics for the spherical projection.

These routines are redundant, not in a particularly reusable form, and application specific rather than general purpose. For example, the routines `ancor.f`, `htrms.f` and `velrms.f` are specific to `nogstat`. The routines `nrancor.f`, `nrhtrms.f` and `nrvelrms.f` are specific to `norstat` (a no longer active modification of `nogstat`). The routine `stats.f` is specific to `verobs`. The routines `differs.f` and `differp.f` are part of yet another application Ocean Model Support Program (OMSP). This component makes them more reusable and general purpose.

C. STATISTICS DATABASE

This component is the data storage for the model verification statistics. It uses

FNMOCC's relational database system, Empress. There is a table for each model and a generic table structure used to create the table structures for each model's table. This database resides on the database workstation. This component allows easy data access and consolidates all the model statistics in a single location. It will have one year's data on line and archive the older data off line. This is to accommodate the model developers' requirements to frequently use the historical data for weekly, monthly, quarterly, seasonal and annual analyses and reports.

D. GRAPHICS

This component creates the graphs of the model statistics based upon the user's requests. The programs are written with one of FNMOCC's graphics software packages, Interactive Data Language (IDL). These programs read the model statistics from a text file. The model statistics are retrieved from the database, formatted and written to a text file for IDL programs. The IDL program creates a GIF file to display in a Hyper Text Markup Language (HTML) page back to the user.

E. USER INTERFACE

This component is the gateway to the model verification system via a web browser. It gives the users the ability to compose their requests for the model verification statistics they want via HTML forms. When the users access the system they are presented with a variety

of on-screen choices. These include a list of models, forecast periods (τ), atmospheric pressure levels, parameters (air temp, wind speed, etc.), statistics, and verification sources. Additionally, they can select the observation types and types of graphs and can also specify the time period they want graphically presented. After they make the various selections and request the information, their request is processed and the information is returned to their screen. Their request is processed dynamically, using a perl script. This is important since it eliminates the need to have any pre-created static GIF files. This is an important capability since static files are less flexible, use greater storage space, and require a great deal of maintenance to keep them current. The perl script parses the user requests, executes the script to query the statistics database and retrieve the data, executes the graphics programs and returns the graphs to the user via a web browser.

F. MEETING THE REQUIREMENTS

Each of the FNMOC Model Verification System components demonstrates the ability to eliminate shortfalls in the current system as well as providing additional enhancements. First, the operational run component computes the statistics twice daily for continuous accumulation of model statistics. It is designed for flexible modification whenever a requirement change occurs via SMS variables and the `namelist` functionality in Fortran 90. Additionally, the statistics database standardizes the data format, data storage, data access and data location by having all the model statistics in a single database. Also, the statistics library standardizes the statistics computation for all the map projections. The text file format

for the intermediate files for the data insertion into the statistics database standardizes how the intermediate files need to be written. This is beneficial in the event these files need to be read directly or if the model statistics from some other systems need to be added into the statistics database. The *graphics* component creates the standardized graphs of the model statistics. The user interface component provides the users an easy-to-use model verification system for all FNMOC models. No special training is required to use the model verification system if users know how to use a web browser. It is a single point and common way to access the model statistics without any knowledge about the system. All these components help to standardize the various aspects of the model verification in FNMOC.

The FNMOC Model Verification System Prototype demonstrates the concept of a single, universal model verification system for all FNMOC models. It provides a vehicle for a better understanding of the environment and requirements problem being addressed. It demonstrates what is actually feasible with the existing technology, and where the technical weak spots still exist at FNMOC. Additionally, it is an effective way to ensure the requirements accurately reflect the user's real needs. The prototype demonstrates to the users what is functionally feasible and provides an analysis test bed and vehicle to validate and evolve the system requirements. [Ref. 17]

III. REQUIREMENTS

A. GENERAL REQUIREMENTS

Generally, the model forecasts are verified against the observed meteorological values or the initial conditions over a set of grid points covering the forecast region called the model's analysis. "V. Bjerknes in 1904 recognized that forecasting is fundamentally an initial-value problem in mathematical physics and, moreover, that the basic system of equations to be solved was already known, at least in general form." [Ref. 10] This research addresses a part of the general requirements, verification against the observational values, but future efforts could expand to address the entire general requirements in the future.

B. FNMOC USER REQUIREMENTS

FNMOC user requirements were obtained through a user survey, review of the current model performance summary reports [Ref. 13], and a meeting with the model team leaders and model verification personnel. The initial user requirements indicated that the models need to be verified against the model's analysis and/or observational data which matches with the general requirements. The verification against the observational data appears to be more widely and frequently used at FNMOC. Therefore, this research concentrated on the verification against observational data initially and will add the verification against the model's analysis in the future. Table 2 below shows the results of the user requirements analysis for

each model. The requirements analysis also showed that bias, standard deviation (stdev) and root mean square (rms) are used more frequently than any other statistical measures. Users indicated that the various model statistics should be stored on-line and immediately accessible for up to one year and off-line for longer period. The users indicated the most widely used graphics to display these statistics are scatter plots, time-series plots and height-series plots.

Table 2. List of user requirements for each models

model	geometry	parameter	levels	taus	stats	verif_source
NOGAPS	global_360x181	air_temp, geop_ht, pres, wnd_spd	sfc, 1000, 925, 850, 700, 500, 400, 300, 250, 200, 150, 100	0-144 at 12 hour increment	bias, rms, std	obs
NOGAPS	global_360x181	air_temp, geop_ht, pres, wnd_spd	sfc, 1000, 925, 850, 700, 500, 400, 300, 250, 200, 150, 100	0-144 at 12 hour increment	anc	anal
NOGAPS	asia_nest1_appl, conus_nest1_appl, europe_nest1_appl, ind_ocn_nest1_appl	air_temp, geop_ht, pres, wnd_spd	sfc, 1000, 925, 850, 700, 500, 400, 300, 250, 200, 150, 100	0-48 at 12 hour increment	bias, rms, std	obs
NOGAPS	europe_nest2_appl2	air_temp, geop_ht, pres, wnd_spd	sfc, 1000, 925, 850, 700, 500, 400, 300, 250, 200, 150, 100	0-48 at 12 hour increment	bias, rms, std	obs
NOGAPS	europe_nest3_appl2	air_temp, pres, wnd_spd	sfc	0-24 at 6 hour increment	bias, rms, std	obs
NORAPS_ASIA	asia_nest1_appl	air_temp, geop_ht, pres, wnd_spd	sfc, 1000, 925, 850, 700, 500, 400, 300, 250, 200, 150, 100	0-48 at 12 hour increment	bias, rms, std	obs
NORAPS_CONUS	conus_nest1_appl	air_temp, geop_ht, pres, wnd_spd	sfc, 1000, 925, 850, 700, 500, 400, 300, 250, 200, 150, 100	0-48 at 12 hour increment	bias, rms, std	obs
NORAPS_EUROPE	europe_nest1_appl	air_temp, geop_ht, pres, wnd_spd	sfc, 1000, 925, 850, 700, 500, 400, 300, 250, 200, 150, 100	0-48 at 12 hour increment	bias, rms, std	obs

model	geometry	parameter	levels	taus	stats	verif_source
NORAPS_IND_OCIN	ind_ocn_nest1_appl	air_temp, geop_ht, pres, wnd_spd	sfc, 1000, 925, 850, 700, 500, 400, 300, 250, 200, 150, 100	0-48 at 12 hour increment	bias, rms, std	obs
COAMPS_EUROPE	europe_nest2_appl2	air_temp, geop_ht, pres, wnd_spd	sfc, 1000, 925, 850, 700, 500, 400, 300, 250, 200, 150, 100	0-48 at 12 hour increment	bias, rms, std	obs
WAM_GLOBAL	global_360x181	sig_wav_ht, peak_wav_per	sfc	24-120 at 12 hour increment for obs, -24 to -120 by 12 hour increment for anal	bias, rms, std	obs, anal
PIPS_N_HEM	n_hem_280x360	ice_cvr, sea_temp	sfc	24, 48, 72, 120 for obs 1,2,3, 5-day old for anal (persistence)	bias, rms, std	obs, anal
OTIS_GLOBAL	global_360x181	sea_temp	dpth_sfc, **	0 for obs 1,2,3-day old for anal (persistence)	bias, rms, std	obs, anal
OTIS_W_ATL	w_atl_211x186	sea_temp	dpth_sfc, **	0 for obs 1,2,3-day old for anal (persistence)	bias, rms, std	obs, anal
OTIS_W_PAC	w_pac_201x201	sea_temp	dpth_sfc, **	0 for obs 1,2,3-day old for anal (persistence)	bias, rms, std	obs, anal
TOPS_GLOBAL	global_360x181	sea_temp	dpth_sfc, ***	24, 48, 72	bias, rms, std	obs

model	geometry	parameter	levels	taus	stats	verif_source
TOPS_W_ATL	w_atl_211x186	sea_temp	dpth_sfc, ***	24, 48	bias, rms, std	obs
TOPS_W_PAC	w_pac_201x201	sea_temp	dpth_sfc, ***	24, 48	bias, rms, std	obs
MISC_GRIDS	n_hem_900x136	ice_cvr	dpth_sfc	0	bias, rms, std	obs
SSM/I_OI	s_hem_900x91	ice_cvr	dpth_sfc	0	bias, rms, std	obs

** 0, 2.5, 7.5, 12.5, 17.5, 25, 32.5, 40, 50, 62.5, 75, 100, 125, 150, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1750, 2000, 2500, 3000, 4000, 5000 meters

*** 0, 2.5, 7.5, 12.5, 17.5, 25, 32.5, 40, 50, 62.5, 75, 100, 125, 150, 200, 300, 400 meters

C. ASSUMPTIONS

Several assumptions were made during this research to focus on meeting the most urgent requirements. First, this research concentrates only on verifying the model forecast against the observational data since more users indicated they would like to verify against the actual data. The assumption here is that the results and methodology can be generalized to the verification versus the model analysis as well. Second, the system is purposely designed to be as independent from FNMOC's relational database management system (RDBMS) Empress as possible by using the text files between the DBMS and the system in the event the DBMS is changed to some other RDBMS. The assumption is that these text files can be interfaced with any RDBMS. Third, WAM_GLOBAL is used as the representative ocean model in the initial system rather than all the ocean models. Other ocean models as well as the meteorological models will be added later. This assumes that the results and methodology can be generalized to other models as well. Fourth, the system only calculates the bias, stdev and rms initially and will add the other statistics later. The assumption is that other statistics can be added.

The system will evolve and mature as the users' inputs from the user prototype evaluations are evaluated. More general verifications will be implemented as the full system is implemented to meet all the requirements identified during the requirements analysis, namely, the verification against the model analysis, the addition of other oceanographic and meteorological models, the addition of statistics (anomaly correlation, probability of detection, threat scores, etc.), consideration of the map factors in the verification against the

model analysis and addition of other graphics for different combinations.

D. LIMITATIONS OF RESEARCH

This research demonstrates the concept of a universal model verification system for all FNMOC models by use of a prototype system. This prototype is implemented to meet the minimum requirements, but provides the capability to expand to meet the additional requirements as well as future requirements. The limitations in the system are listed in the following paragraphs.

The model statistics is computed for different observation types separately rather than combined. The map factors are not considered in the verification against the observations. In other words, when the model forecasts are verified with respect to the observational data, all the latitude/longitude points are considered equally weighted. This system creates only the time-series plots to narrow the scope of the graphics component. Other types of graphics can be added in the future.

The prototype is bound to the current FNMOC operating systems and software since it should use the FNMOC environment. Therefore, this system may not be as portable as it could be when there is a significant change within the FNMOC environment, such as a different operating system, new DBMS, new graphics software, new standard shell, etc. Some of the lower level modules that interface to the ISIS latitude/longitude/time (LLT) data have unavoidable coupling to ISIS due to the different observational data type structures.

IV. DESIGN

The initial prototype system design has narrow scope to concentrate on meeting the immediate and minimum requirements. Additional requirements will be added in future versions to test the system's flexibility. The immediate requirements are verification of the NOGAPS, NORAPS, COAMPS and WAM_GLOBAL models after the model runs for daily, weekly, monthly and quarterly reports. The prototype will present bias, stdev and rms on a few specified parameters (e.g., air temp and wave height). The WAM_GLOBAL model is used as a representative ocean model in the prototype.

The prototype of the initial system addresses these requirements to prove the design and implementation concept. Additionally, this will demonstrate the feasibility of using a single universal model verification system for FNMOC. "The system need not have a finely tuned prototype before it is implemented. In fact, one merely asks of the prototype that it contains a flexible set of hardware and software tools for continuous redesign, and have access to real-time data (with standardized formats)." [Ref. 15] The prototype has a GUI to allow the users to view the model statistics in graphical form on a web browser.

The prototype is developed and implemented in FNMOC's operational environment to ensure that it is useful in that environment. "The important thing is that the development takes place primarily in the operational office, with the direct involvement of the people who are going to use it." [Ref. 15]

A. OVERALL STRUCTURAL DESIGN

Figure 2 shows the overall structural design of the FNMOC model verification system. It shows the five main components described in Chapter II and their relationships.

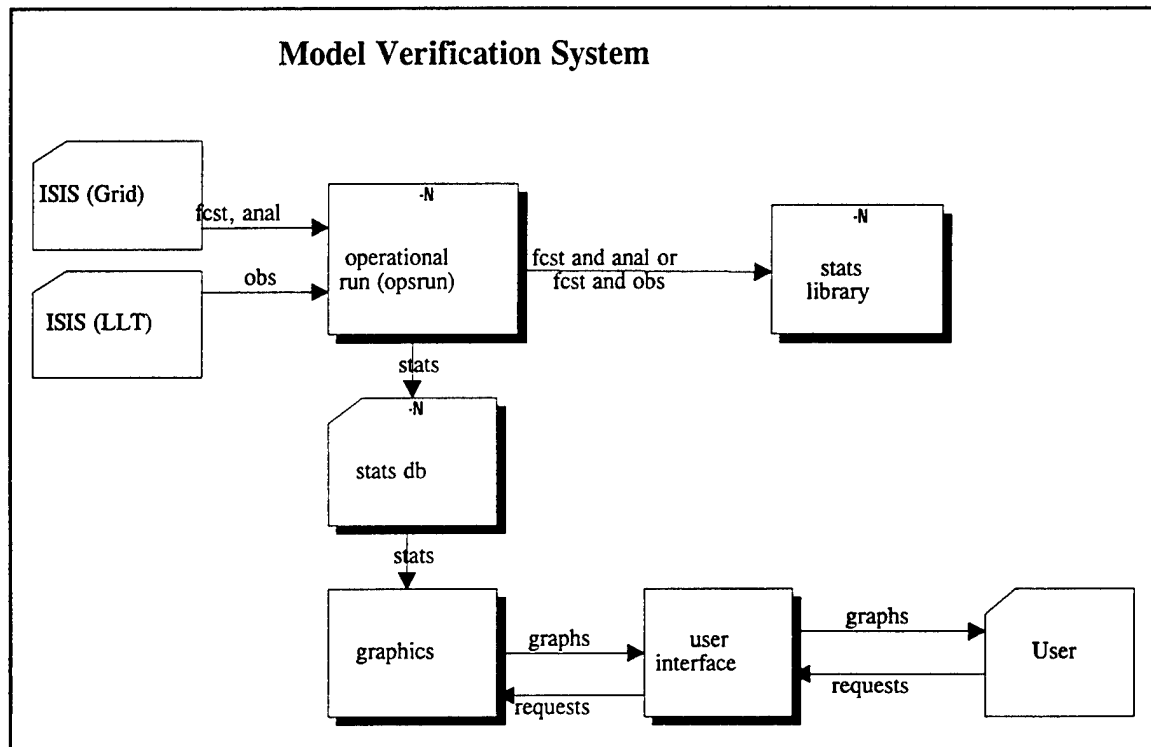


Figure 2. Level 1 Diagram

The *operational run* component computes the model verification statistics twice a day by executing either *verobs* or *veranal*. *Verobs* reads the model forecasts and analysis from the ISIS grid database and observational data from the ISIS LLT database. *Verobs* and *veranal* use the statistics computation routines in the reusable *statistics library* component to perform the computation. They write the statistics in the text files in the standardized format (one for each model) for transfer to the FNMOC database development workstation

where the *statistics database* component resides. The *operational run* component then inserts the model statistics into the database. The transfer and the database population is not done twice a day by FNMOC operations currently as part of the ops run because operations is not one of the owners of the *statistics database* component. Currently, the transfer and database populations are done by FNMOC staff about once a day during the week days. This is because the *statistics database* is not part of ISIS currently, and the FNMOC ops run does not write to any non-ISIS databases. The *statistics database* should become a part of ISIS after the prototype evaluation. The *user interface* component presents HTML forms to users which they use to make their requests. This component processes the user requests by retrieving the model statistics from the *statistics database* and formats the data for the *graphics* component. The *graphics* component then creates the graphics to send to the users. The following sections describe the detailed design for each module in each of the five components. The source code for all the modules is in the Appendix B.

B. DESIGN OF OPERATIONAL RUN (OPS RUN)

The ops run has two korn shell scripts, *mverif.job* and *statupd.ksh*. It also has Fortran programs *verobs.f90* and *veranal.f90* (in the future) along with a Fortran header file *v_data.h*. *mverif.job* reads the SMS variables and executes the appropriate program based on the SMS variable VERIF_SOURCE value. FNMOC operations executes this job script to compute the model verification statistics twice a day after each model run is completed. The model statistics are written to a text file for each model, geometry and date-time-group

(dtg). *Statupd.ksh* transfers the model statistics text files from one computer to another workstation, formats the text file and populates the statistics database using Empress Standard Query Language (SQL). Figure 3 shows the design of ops run graphically.

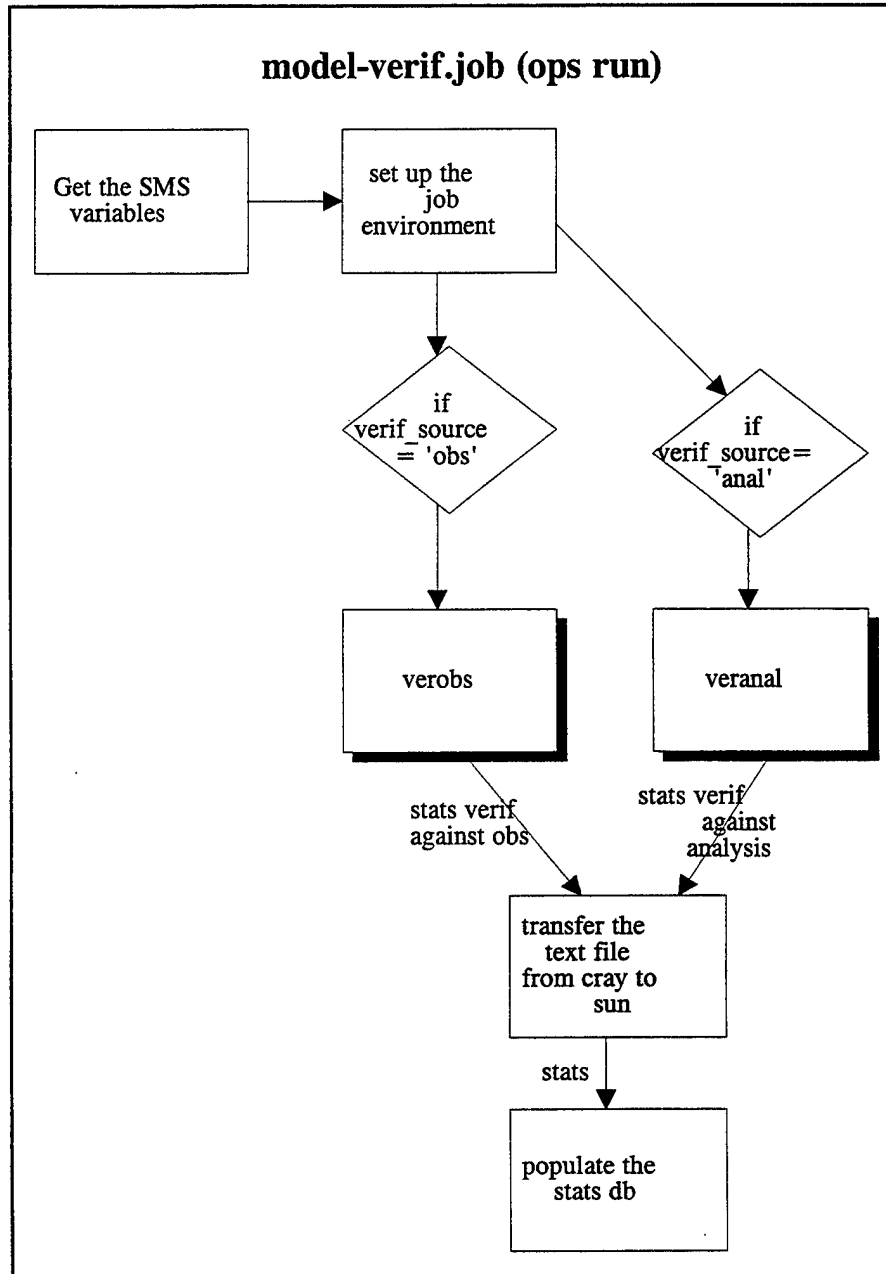


Figure 3. Ops Run Diagram

1. Design of *mverif.job* (Cray)

This system needs to be flexible to incorporate new requirements and changes to the existing requirements. The use of SMS environment variables in *mverif.job* is one of the methods this system uses to assure this flexibility. FNMOC uses SMS to run the operational runs (ops runs) and uses the SMS environment variables extensively to control the ops runs. The SMS variables are set before the *mverif.job* is executed and passed into the job. Figure 4 shows the list of the SMS variables and their description. By changing the value of these SMS variables, *mverif.job* is executed to verify all the FNMOC models without changing the job itself. The values for these SMS variables were set based on the user requirements analysis shown in Table 1. The complete SMS value list for each model is included in the Appendix A.

CRDATE - watch date-time-group (e.g., 1998013112)
ISIS_TABLE - ISIS table name, coincides with the model name (e.g., NOGAPS, WAM_GLOBAL)
GEOMNM1 - geometry name defined in ISIS (e.g., global_360x181)
GEOMNM2 - second geometry name
TAUI - beginning forecast time
TAUE - ending forecast time
TAUINC - increment of forecast time
VERIF_SOURCE - either 'obs' or 'anal'
OPSBIN - directory in which the binaries for the operational runs reside
ISIS_INIT - ISIS initialization script name
PROGBIN - directory in which the binary for this system resides, may be same as OPSBIN

Figure 4. SMS Variables and Description

mverif.job is a `korn` shell script that FNMOC operations execute to compute the model verification statistics twice a day. This job performs various operational job required tasks such as setting up the operational job environment and executing the correct ISIS initialization script. It reads the required SMS variables and exports them to the subsequent shells. It creates the appropriate sub-directory with the month (mmm, e.g., mar) and year (yyyy, e.g., 1998) from CRDATE to write the model verification statistics. There are separate sub-directories for each month and year. It copies the model's namelist files to the \$TMP directory and executes the appropriate executable file based on the variable VERIF_SOURCE's value as shown in Figure 3. If the value of VERIF_SOURCE is 'obs,' *verobs* is executed and if the value is 'anal,' *veranal* is executed. It then writes the resulting model statistics into the standardized text file in the appropriate data directory (/a/ops/etc/dynamic/app/mverif/mmm/yyyy). It finishes the process by completing the job accounting information and writing the joblog file to the operational joblog output directory (/a/ops/job/out).

2. Design of statupd.ksh (Sun)

This is a `korn` shell script that inserts the model statistics into the appropriate data table in the statistics database, *stat_db*. It is executed after *mverif.job* is finished. It formats the transferred text file using the `awk`. *Statupd.ksh* determines if new statistics files exist in the data directory. It also determines the database table to populate based on the model name in the statistics files. It then inserts the statistics into the appropriate data table in *stat_db*.

3. Design of Fortran Include File, V_DATA.H

All data definitions and constants used throughout the Fortran 90 programs (*verobs*, *veranal*) and their subroutines are in the Fortran include file called *V_DATA.H*. This provides for easier maintenance since the modification is isolated in one location rather than in multiple locations when changes in a variable occur.

4. Design of VEROBS.F90, The Main Verification Against Obs Program

This is the top level Fortran program that verifies the model forecasts with respect to the observational data for various models. *Verobs.f90* uses the namelist functionality. This helps to achieve flexibility in addition to the usage of the SMS variables. Like the SMS variables, namelist values are set external to the program. Each model can have a namelist file for the surface level and a namelist file for the upper levels. The namelist files are explained in more detail in subsection *a* below. Figure 5 shows the structural design of *verobs* module.

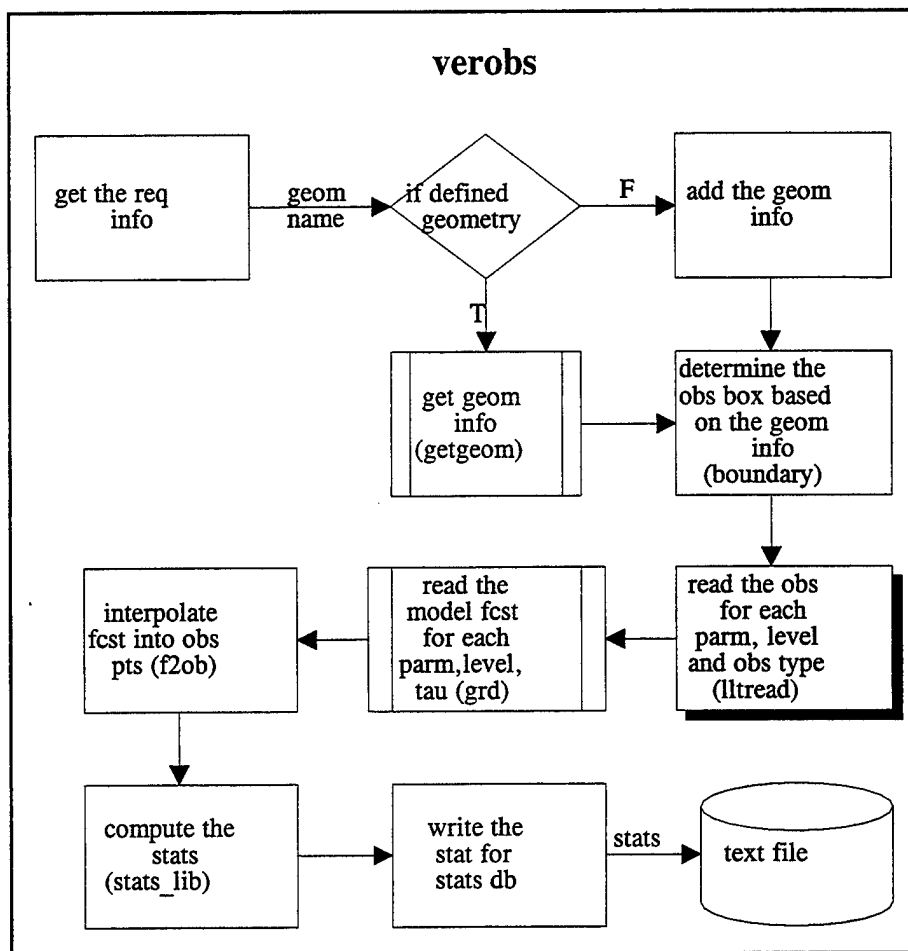


Figure 5. Verobs Design

Verobs.f90 reads the SMS environment variables described in the *ops run*. It determines if the geometry in GEOMNM1 is defined in the ISIS. If it is not defined, the geometry information is added. Once the geometry is defined, it determines the earth area to read the observations for the model verification. It then reads the model forecast from the ISIS grid database and observations from the ISIS LLT database. It interpolates the model forecast to the observational data points to compute the bias, stdev and rms using the

statistics library component. It writes the statistics to a text file. The text filename is determined by the SMS variable values for ISIS_TABLE, GEOMNM2 and CRDATE. The modules (Fortran subroutines) called by *verobs* are described in the following sections starting in section *b*.

a. Namelist file format for each model

Each model can have two namelist files, one for the upper environmental levels and one for the surface level. The second namelist file is needed because the surface level is different for different parameters and level types while the upper levels are common levels for all the parameters. For example, the surface level for air temperature is 2.0 meters, while the surface level for wind speed is 19.5 meters. Figure 6 shows the list of the namelist items and their descriptions. The complete namelist files for each model are included in Appendix A along with Table 3. Table 3 shows the namelist items specified by the users for each model.

parm - the list of model parameters to verify dsetname - the data set name for the model forecast parameter in ISIS grid data obs_parm - the corresponding parameter in the observational data in ISIS LLT data odsetname - the data set name for the observational parameter in the LLT data units - unit of the given parm lvltype - level type of the given parm obstype - the observational type to read from LLT data level - vertical level to read stat - type of statistic to compute (bias, std, rms) tval - threshold value for the threat score, probability of detection, false alarm, and skill score; it will be an optional item since it is not always be used.

Figure 6. Namelist Variables and Description

For each value of parm, there must be a corresponding value for dsetname, obs_parm, odsetname, units, lvltype and obstype. For example, if the parm value is 'air_temp,' there must be values for the dsetname which can be 'fcst_ops,' the obs_parm which should be 'air_temp,' the odsetname which can be 'fnmoc,' the units which should be 'deg_K,' the lvltype which can be 'isbr_lvl' and the obstype which can be 'raob_qc.'

b. boundary

This Fortran subroutine determines the geographical area from which to extract observations to use in the verification. It computes the minimum and maximum latitude and longitude for a given geometry's area box. These minimum and maximum latitude and longitude are required for reading the observational data from ISIS llr database. It fills the single dimension arrays, x and y, with the values representing the left, right, bottom and top boundaries. It converts x/y values to latitude/longitude values by calling FNMOC utility, vxyl1. It then finds the minimum and maximum latitude and longitude.

c. f2ob

This Fortran subroutine interpolates the model forecast fields to the observation locations. It obtains the number of rows and columns for the geometry defined by GEOMNM2 by the ISIS utility `getgeom` and finds the maximum array dimensions for x and y arrays. It converts observations latitude/longitude points to x/y values by calling FNMOC utility, `vllxy`. It then interpolates the model forecast to the observation points by calling FNMOC utility, `finterp`.

d. intgeom

This Fortran subroutine interpolates one geometry to another geometry. This interpolation is necessary when a model needs to be verified in the geometry other than the geometry in which it is stored. For example, the global models such as NOGAPS are compared to the regional models such as NORAPS_ASIA in the same Asia region. It gets the information on the two geometries defined by GEOMNM1 and GEOMNM2 by calling the ISIS utilities `ggrd` and `getgeom`. It then interpolates the first geometry to the second geometry points by calling the FNMOC utility `chgeom`.

e. lltread

This Fortran subroutine calls the appropriate LLT read subroutine based upon the observation type. This subroutine acts as the middle man between the main program `verobs` and the `llt` read subroutines for each observation type. There are separate LLT read modules for each observation types because the include files and the data structures in ISIS are different for different observation type. For example, the observation type 'raob_qc' is read by the `raob_qc_read` subroutine. The subroutine `raob_qc_read` uses two ISIS includes

files, `common.inc` and `RAOB_QC.H`. It has to use the data structure `TYPE (raob_qc_int)` and `TYPE (raob_qc)`, and use the 'raob_qc' sequence type in the call to the ISIS LLT read utility `lrd`. Currently, *verobs* reads only the observation types, `raob_qc`, `sfc_lnd`, `sfc_ship`, `sfc_ship_met_qc` and `alty` using *raob_qc_read*, *sfc_lnd_read*, *sfc_ship_read*, *sfc_ship_met_qc_read* and *alty_read*, respectively. Other observation types will be added as more models and parameters are added to the model verification system.

C. DESIGN OF STAT LIB

The requirements analysis indicated the following statistics should be included in the initial prototype: Mean error (bias), standard deviations (stdev), root mean square (rms), anomaly correlation (anc) for most of the model parameters and threat score, probability of detection, false alarm, and skill score for selected model parameters. This research focused on just the first three statistics since they are most often used. The other statistics will be added in future versions. Some of the existing routines will be modified to make them more general purpose and added as part of the statistics library. The design of each statistics module is described in the following sections.

1. Map Factors

The basic formula for the statistics are the same, but the statistics routines need to take the map projections into account to compensate for differences in earth area at different latitudes on the earth. This is achieved by applying a map correction factor at each grid point. Then the routines calculate the area using the new x- and y- coordinates. The general

mathematical formula for the statistics is

$$\int_{\text{surface}} s(\lambda, \varphi) \delta a / \text{surface area} = (\sum s_n W_n) / \sum W_n$$

where s =value at a given i/j grid point, λ =longitude, φ =latitude, W_n are weighting factors.

There are two distinct cases to consider in the weighting factors or map factors. First, the observations are all independent. Therefore, each observation has equal weight, and the statistics formula becomes a simple average formula. Second, the grid points in which they lie must be weighted according to the relative size of the physical area related to the map factors for the relevant projection. In general,

$$W_n = (\Delta x_{ij} \Delta y_{ij})_n$$

where $\Delta x_{ij} = h_x \Delta \lambda_{ij}$, $\Delta y_{ij} = h_y \Delta \varphi_{ij}$, h_x and h_y depend on the map projection and $\Delta \lambda$ is the difference between two longitude, and $\Delta \varphi$ is the difference between two latitude of the box the s value lies.

According to FNMOC's model meta data database table, `ops_meta_grid_db.grid_reg_geom`, FNMOC's models use polar stereo, spherical, lambert, and mercator projections. FNMOC verifies NOGAPS in spherical projection against its analysis with the map correction factor currently. Some models are verified without the map projection consideration against the observational data with the assumption that each observation has the equal weighting factor of 1. The map factors for each map projection used at FNMOC are as follows: [Ref. 10, Ref. 12]

Spherical:

$$h_x = \cos \varphi_{ij}$$

$$h_y = 1$$

Polar Stereo:

$$h_x = h_y = (1/2)(1 + \sin\phi_{ij})$$

Mercator:

$$h_x = h_y = \cos\phi_{ij} / \cos\phi_0$$

where ϕ_0 is the latitude at which the projection is “true” and can be obtained from FNMOC database attribute geom_parm_1 in degrees.

Lambert:

$$h_x = h_y = (\cos\phi_{ij}/\cos\phi_1)^{1-K} [(1+\sin\phi_1) / (1+\sin\phi_{ij})]^K$$

$$\text{where } K = \ln(\cos\phi_1/\cos\phi_2) \div \ln[\tan(\pi/4 - \phi_1/2) / \tan(\pi/4 - \phi_2/2)]$$

ϕ_1, ϕ_2 are standard latitudes of the projection; their values can be obtained from FNMOC database attributes geom_parm_1 and geom_parm_2 in degrees.

2. Bias (Mean Error)

“Error is the simple difference of forecast minus verifying analysis or observation. The difference (error) field provides a quick look at a model’s forecast performance or bias. Bias or tendency describes whether a synoptic field or feature is under or over-forecast.” [Ref. 13] The advantage of the simple difference fields is that they are easy to compute and understand. They provide a quick look at the model forecast performance. [Ref. 13] The formula used for bias is

$$\sum(F_n - O_n) W_n / \sum W_n \text{ where } F \text{ is forecast, } O \text{ is the observation.}$$

3. Standard Deviations (stdev)

“Standard Deviation (stdev) is a measure of the scatter or variability about the mean in a series of observations. Standard Deviation is the positive square root of the variance.” [Ref. 13] The formula used for stdev is

$$\text{sqrt} [(\sum((F_n - O_n)^2)W_n) / \sum W_n - (\sum(F_n - O_n)W_n / \sum W_n)^2].$$

4. Root Mean Square (rms or rmse)

“Root Mean Square Error (RMSE) is defined as the positive square root of the mean square error (MSE). MSE is the mean square of any residual. RMSE is the also called the standard error of estimate.” [Ref. 13] “The RMSE is a quadratic score that gives the average magnitude of the errors. This statistics gives more weight to large errors than to small errors in the average, and is useful when large errors are undesirable.” [Ref. 16] The formula used for rms is

$$\text{sqrt} ((\sum(F_n - O_n)^2 W_n) / \sum W_n) .$$

D. DESIGN OF STATISTICS DATABASE

The *statistics database* component uses FNMOC’s RDBMS, Empress, to consolidate all the model statistics in one location. The database has a table for each model. The data in the tables are based upon the model developers’ recommendations. A consolidated database will speed up the data retrieval and insertion. It will also accommodate modifications if the table structure needs to be modified for a particular model in the future. Figure 7 shows the SQL command used to create the generic table *model_stats*.

```

CREATE TABLE MODEL_STATS
(verif_date      character(10,1)    not null,
sample_size     integer             not null,
parm_name       character(32,1)     not null,
unit_name       character(32,1)     not null,
geom_name       character(32,1)     not null,
lvl_type        character(24,1)     not null,
level_1         float(2)            not null,
tau             integer             not null,
stat_type       character(16,1)     not null,
stat_value      float(2)            not null,
verif_source    character(8,1)     not null,
obs_type        character(24,1)     not null)

Indices: NORMAL (2, 15) parm_id ON (parm_name)
         NORMAL (2, 15) geom_id ON (geom_name)
         NORMAL (2, 15) level_1_id ON (level_1)
         NORMAL (2, 15) tau_id ON (tau)
         NORMAL (2, 15) stat_type_id ON (stat_type)
         NORMAL (2, 15) obs_type_id ON (obs_type)

```

Figure 7. SQL used to create the generic table

The basic table attributes are shown in Figure 8. This structure was saved in a file *stattbl* by the Empress command '*display model_stats all dump into stattbl;.*' The tables for other models were created using the structure stored in the file *stattbl* by the Empress command '*create nogaps from stattbl;.*'

The use of the indices improved the retrieval performance from greater than 3 minutes to less than 5 seconds. However, the insertion performance was degraded especially as the number of records increased. The tradeoff between these two scenarios favored the increase in the retrieval performance. The compromise may be to drop the indices, perform the insertion, then rebuild the indices every time data are updated according to a local

database expert, but this compromise has not been tried for this research.

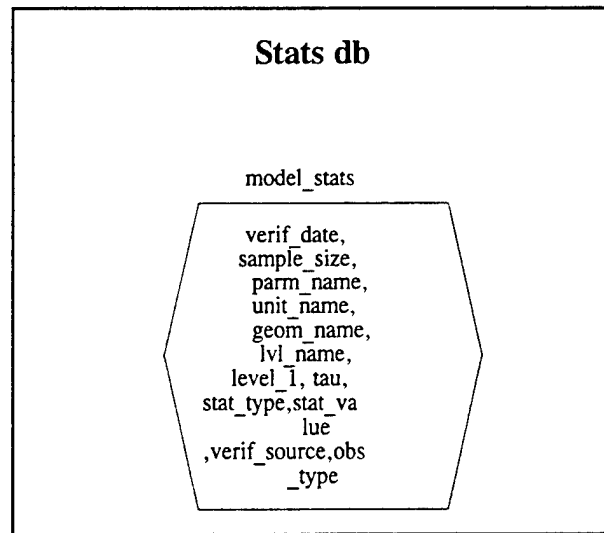


Figure 8. Statistics Table Attributes

E. DESIGN OF GRAPHICS

The *graphics component* consists of one IDL program currently in the prototype. This IDL program plots the bias, stdev and rms for a single model, single geometry, single parameter, single forecast period, single atmospheric pressure level and single observation type on one graph. More IDL programs will be added in future enhancements to create the graphs for other combinations. The IDL program creates a GIF file for display in web browsers. Figure 9 shows the structural design of the *graphics component*.

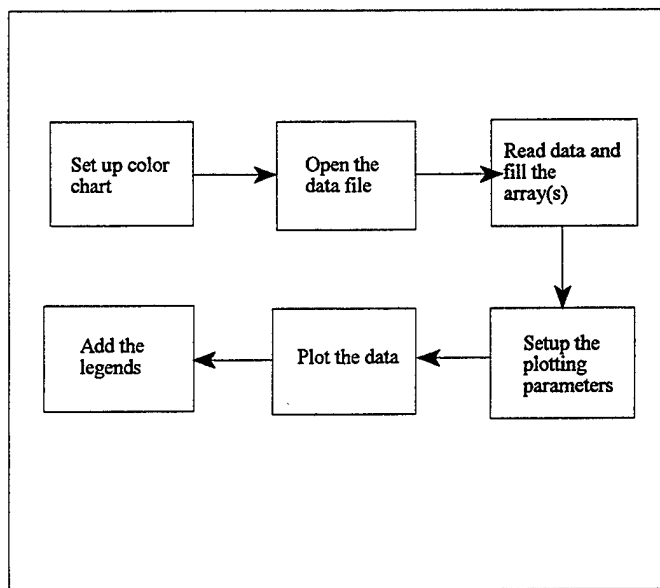


Figure 9. Graphics Design

The IDL program reads the data from a text file into the IDL array. It reads various environment variables to use within the program. It builds sub-arrays based upon the statistics type, e.g., bias, stdev and rms. Future IDL programs will build different sub-arrays based on the models, geometries, forecast periods, atmospheric pressure levels, or observation types. It formats FNMOC's 10-digit date-time group (1998032000) into a more meaningful date-time format (00Z 20Mar 98) to use on the x-axis label. It plots the bias and rms with different symbols and colors, and stdev as the shaded areas above and below the bias as requested by the users. Future IDL programs will create the scatter plots and other plots as the requirements change. An example of the current graphs is shown in Figure 13 in the next section.

F. DESIGN OF USER INTERFACE

Figure 10 shows the structural design of the *user interface* component. This component ties together the *statistics database* and *graphics* components. It has a GUI and Common Gateway Interface (CGI) to give the users access to the model statistics.

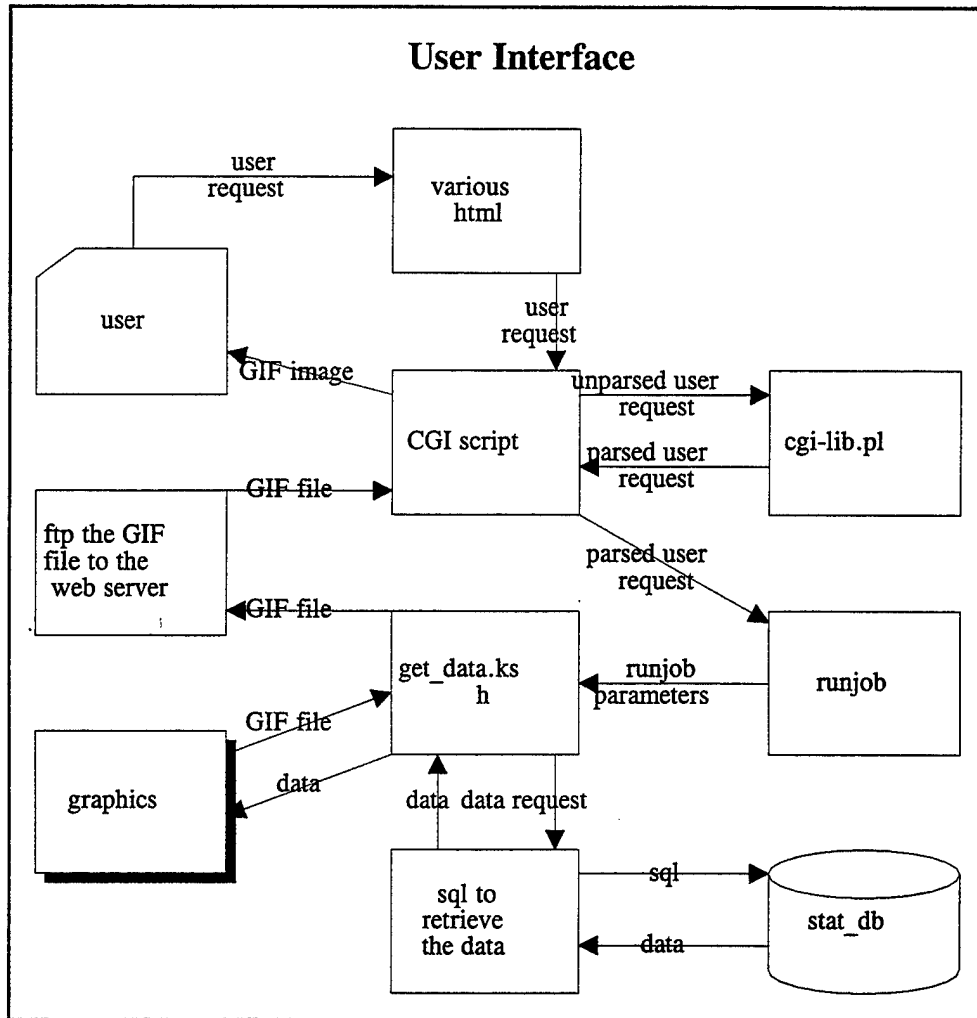


Figure 10. User Interface Design

The GUI portion has several html forms to interface with the users. The first model verification system web page shown in Figure 11 is created by the file *index.html*. It is shown using a Netscape web browser.

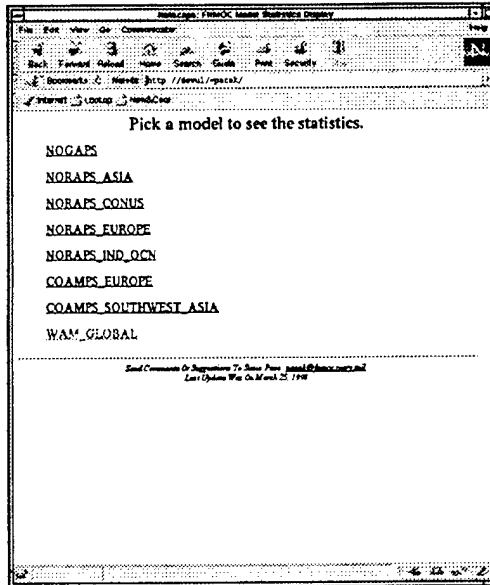


Figure 11. Home Page

When the user picks a model by clicking on a selection button, the model page is displayed. An example model page is shown in Figure 12 for the WAM_GLOBAL model. This page is created by the file *wam.html*. The model pages have the default values already selected, but the users can change the values by simply clicking other choices. The user would then click on the 'Submit the query' button when (s)he is finished composing the request or the 'Cancel' button at any time.

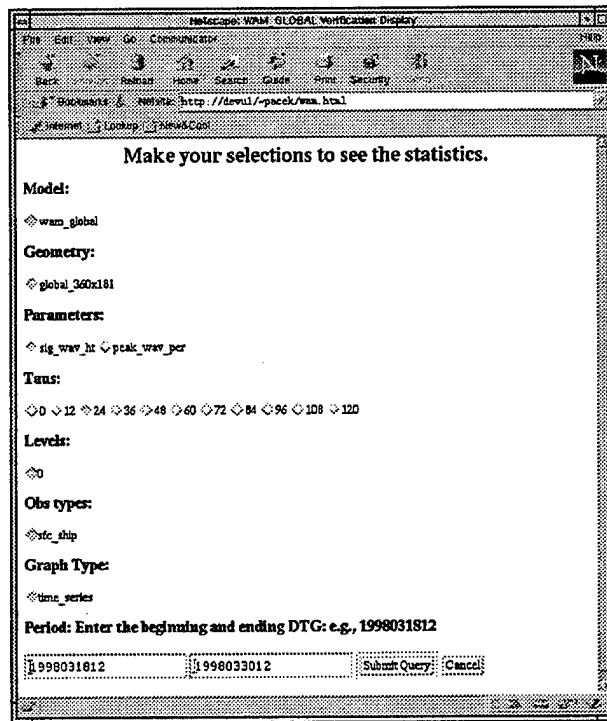


Figure 12. WAM_GLOBAL Statistics Page

Once the user submits the query, the CGI portion of this component takes over.

The CGI portion has several programs to satisfy the user requests dynamically. Since this research was implemented in the current operational environment, several accommodations had to be made. The FNMOC intranet web server is on a Sun workstation called 'devu1.' As discussed earlier in the *statistics database* component, the stat_db is on a database development workstation called 'div60-3.' The model statistics must be retrieved from div60-3 and transferred to devu1 to work in the current configuration. The GIF images are also created on div60-3 because it has better performance than devu1.

When the user's request is submitted, a CGI perl script in the cgi-bin is executed to processes the user request. This CGI script parses the user request and starts the

processing of the user request via runjob on div60-3. This runjob involves data retrieval from *stat_db* by Empress SQL. It then formats the retrieved data by awk for IDL to read. It executes the IDL program described in the previous section which creates a GIF file. It then transfers the GIF file to the web server via ftpbatch. Once the GIF file transfers successfully, the GIF image is displayed on the web browser.

Figure 13 shows the result of the query from Figure 12 for WAM_GLOBAL. At this point, the user can print the image, save the image in different formats supported by the web browser (e.g. GIF, postscript, etc.) or just view the image.

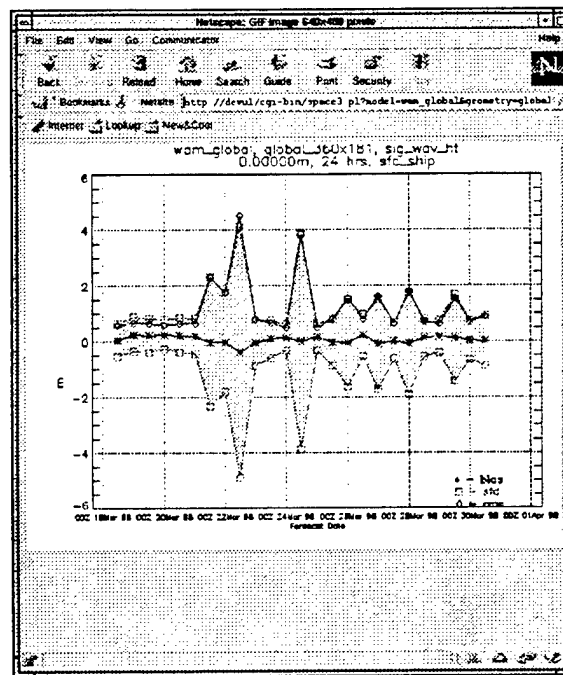


Figure 13. WAM_GLOBAL Statistics Graph

V. EVALUATION

The evaluation of the model verification system included alpha testing and beta testing. The alpha testing was performed by the developer by comparing the computed statistics to the existing verification system *verobs*. The beta testing was performed by FNMOC operations and FNMOC model developers (users). The operations ran the model verification system and *verobs* in parallel for four weeks. The users used the prototype to access the verification statistics via the Netscape web browser. The addition of a new model, `COAMPS_SOUTHWEST_ASIA`, demonstrated the ease of extending the model verification system to include other models.

A. ALPHA TESTING

Each unit (program and subroutines) of the *ops run* components was unit tested using the Fortran debugger, TotalView, which enabled the developer to test the binaries interactively. The debugger allows the tester to assign values to variables, print values of variables, and step through the statements in the calling program and called subroutines. This testing was useful in identifying code-level bugs, e.g., the parameter orders in the actual and formal parameter lists of the subroutine call to *boundary* in the main program *verobs* did not match.

The statistics computed by the *statistics library* subroutines were compared to the statistics computed using Microsoft Excel. A set of numbers from the model forecast

and a set of numbers from the observational data were assigned to two arrays. The statistics bias, stdev, and rms were computed by the *statistics library* subroutines and Excel. These two sets of statistics matched as shown in Appendix C. The test program for the *statistics library* subroutines, *stattest*, is included in Appendix C.

The developer compared the model forecast and observational data at the first 15 latitude/longitude points of the test run of the model verification system to the first 15 latitude/longitude points of the operational verobs to ensure they matched point-by-point as well as the overall computed statistics. An error in the loop index in the statistics library routines was discovered and corrected during the alpha testing.

The test runs included verification of each model in its own area and the verification of the global model in the regional model areas. The purpose was to test the interpolation involved when a model was verified in an area other than its own. The global model NOGAPS was verified in the regional model NORAPS areas (Asia, Continental US, Europe, Indian Ocean) and in the COAMPS area (Europe). This allowed the comparison of multiple models in the same geometry. A change in one of the COAMPS geometries was discovered during the alpha testing. The COAMPS model developer modified the geometry for the model, but the verification systems were not aware of this change. The change in the geometry resulted in the job running without computing any statistics. This change was implemented in the model verification system by changing a SMS variable.

The *statistics database* component was tested by inserting data and performing queries against the database. The time required to return data from the initial queries was

much longer than anticipated. Modifications to the database tables were made which improved query performance significantly.

A set of data from *stat_db* was saved in a text file to be the test data for the IDL program in the *graphics* component. The *graphics* component was tested by printing out the data, printing out the subarrays created from the data and examining the graphs of the model statistics. The *user interface* component was tested using the Netscape web browser. The *cgi* scripts were installed on the FNMOC intranet web server *cgi-bin* because the current web server configuration does not allow the execution of the *cgi* scripts in *cgi-bin* directory of individual users/developers. The FNMOC intranet webmaster installed the *cgi* scripts in the web server *cgi-bin* directory.

B. BETA TESTING

The purpose of the beta testing was to compare the model verification system to the existing verification process and to test the user interface of the prototype. The beta testing was performed by operations and the users at FNMOC. The operational beta testing started on 18 March 1998 by FNMOC operations to verify NOGAPS, NORAPS, COAMPS and WAM_GLOBAL models in their own areas. The verification of NOGAPS in the NORAPS and COAMPS areas was added on 25 March 1998. The model verification system was comparable to the existing verification system *verobs*, but it was much easier to add the verification of NOGAPS in the NORAPS and COAMPS areas.

An important part of the beta test was the evaluation of the prototypes ease of use. Generally, users indicated that the system was very easy to use. Another area examined was how easy it was to access the model statistics. Again, users evaluated this area positively. Another area evaluated was whether the system provided the required displays. For the most part this area was positively evaluated. However, the users wanted more types of graphs such as multiple models on a single chart, wanted the number of observations displayed, and better performance, i.e., faster return of the images. Based on the Beta testing results, the prototype meets the easy-to-use model verification system requirements.

C. MODIFICATION

A new model, COAMPS__SOUTHWEST_ASIA, was added to the prototype after the initial product implementation. This provided an opportunity to evaluate how easy it would be to modify the verification system. It was very easy to add the additional model to the model verification system and no design changes were required. Adding this new model required only a few modifications. The operations executes the job script *mverif.job* with appropriate SMS variables for the model (See Appendix A) after COAMPS__SOUTHWEST_ASIA completes its run. A table for COAMPS__SOUTHWEST_ASIA was created in the *stat_db*. The html forms were updated to reflect the additional model. This demonstrated that the ease of modification requirement was satisfied.

D. SUMMARY

This research has resulted in the development and demonstration of an easy-to-use model verification system for all FNMOC models. Furthermore, it has demonstrated that model verification can be standardized in FNMOC. The model verification standardization was achieved by the several components of the model verification system. The reusable statistics library standardized the statistics computation, the statistics database standardized the data format, data location, data schema, interface to the database, and the graphics standardized the graphical display of the model verification statistics. Additionally, it proved the concept of a easily modified verification system. Adding a model to the system required no design changes and no source code level changes other than adding new modules to handle the additional observation types. Finally, the prototype demonstrated that an easy to use system could be developed to return data and graphs to the user's desktop. The use of a GUI and a web browser provided the users with an easy-to-use access to the model verification statistics. No user training was required. Therefore, the research successfully addressed the research questions and has proven the concepts can be implemented in the full model verification system.

VI. RECOMMENDATIONS AND FUTURE RESEARCH

This chapter discusses the recommendations for future FNMOC model verification systems implementation based upon the results of this research. It suggests areas of future research in fully implementing a single model verification system for FNMOC.

A. RECOMMENDATIONS

FNMOC needs to use one verification system to verify all the meteorological and oceanographical models. This single system will provide many benefits to FNMOC. First a single verification system will eliminate the redundancies and inconsistencies of multiple verification systems. Additionally, it will eliminate the maintenance-intensive verification systems needed to maintain data consistency and accuracy. It will also standardize the model verification process. An additional benefit is that by using the same verification system the model developers will be able to focus more on the model development effort than on the verification process. This research proved the feasibility of using a single model verification system for all the meteorological and oceanographic models at FNMOC by providing a easily modified system. This model verification system demonstrated it can verify all the models and standardize the model verifications at FNMOC and became useful to the model verification/validation group in FNMOC. This model verification system also provides a verification system for the models that do not currently have a verification systems.

FNMOC can provide users with an easy-to-use tool to access the model verification

statistics that meets the users' needs. Having an easy-to-use tool that is readily assessable via the web technology is important for both internal and external users. The internal users could be the model developers, model researchers, managers, statisticians, or others in the command responsible for FNMOC data accuracy. The external users could be US Navy organizations, US Air Force organizations, or other organizations. This research proved that the concept of an easy-to-use model verification system via GUI and the web browsers is feasible. It provides access to the verification statistics and graphs to anyone who has a web browser. Giving users easy access to the model verification statistics makes their efforts to fine tune the models and/or evaluate their performance more effective. It is much too difficult and time consuming to access the information in the currently existing model verification systems in FNMOC.

FNMOC needs to document the model verification requirements for all the models. This research identified and documented each model's verification requirements. This will be useful for current and future model developers and model users both within FNMOC and outside FNMOC.

FNMOC should use prototyping as much as possible in developing new systems. The prototype was a very useful communication tool between the designer/developer of the model verification system and the users. It helped to firm up the requirements and identified additional requirements that were not identified in the initial requirements analysis, especially in the graphics area.

Finally, FNMOC should build a comprehensive model verification/validation system for all the models in FNMOC. The first step in this effort should be building a comprehensive

quantitative model verification/validation system that has three subsystems. One subsystem should display the model forecast and analysis, another subsystem should display the observational data, and a final subsystem should display model verification statistics. This research dealt with the third subsystem. Each subsystem should have the same look and feel interfaces with a GUI. Addition of subject matter expertise (qualitative aspect) to aid the interpretation of the quantitative measures will provide a complete model verification/validation system for all the models at FNMOC.

B. FUTURE RESEARCH

There are many potential extensions for the model verification system to implement all of the user requirements identified during the requirements analysis.

1. Ops Run

First, the new *verobs* should be completed by adding more observation types. This means adding the Fortran subroutines to read the additional observation types from the ISIS LLT database.

Second, more models should be added to the model verification system. There are several additional ocean models to verify, and these ocean models were prioritized for their implementation order. The observation types MCSST, Buoy and Bathy need to be read by the subroutines *mcsst_read*, *buoy_read*, *bthy_read*, respectively, for the OTIS_GLOBAL, OTIS_W_ATL and OTIS_W_PAC models. The next models to implement are TOPS_GLOBAL, TOPS_W_ATL and TOPS_W_PAC using the same observation types as

OTIS models. The users want to compare the OTIS and TOPS models and determine whether FNMOC needs to run both models. The observation type SSMI_EDI_ICE needs to be read by the subroutine *ssmi_edi_ice_read* for the PIPS_N_HEM model.

Third, the verification against the model analysis, *veranal* should be added. This involves several steps. Complete the design, using the existing verification system *nogstat* if feasible. Expand the *statistics database* and *user interface* components to reflect the *veranal* statistics.

2. Statistics Library

The map factor subroutine in the *statistics library* component should be revisited and completed for all the projections used at FNMOC as described in the design chapter because the map factors will be used in *veranal*. Other statistics computation modules should be added to compute the additional statistics such as anomaly correlation, probability of detection, threat score, etc.

3. Statistics Database

The *stat_db* should be incorporated into FNMOC's database and FNMOC database administrators (DBA) should be responsible for administering the *stat_db*. This will allow the operations to write directly into the database as the statistics are computed twice a day. This will eliminate the current method of one person transferring and populating the database manually. This will also make the latest statistics available to the users.

4. Graphics

The graphics component should be expanded by adding more types of graphs. There are many different possible graphs based on the combination of models, forecast periods,

levels, observation type and statistics. Some of the combinations of interest include multiple statistics for a single model, single forecast period, single level, single observation type; multiple levels for a single model, single forecast period, single observation type, single statistic; multiple forecast periods for a single model, single level, single observation type, single statistic; multiple models for a single forecast period, single level, single observation type, single statistic. One of the users suggested adding the maps of the geometries used in the various FNMOC models and adding the number of observations on the graphs. These are very good suggestions and should be added to the system.

5. User Interface

The user interface component should be expanded to handle the various types of graphs described in the previous paragraph. The `html` forms should be made to be more robust and remove the possibility of user input errors. Currently, there is room for user input errors in the `html` forms in matching the atmospheric levels, observation types and parameters because all the levels and observation types (both surface and upper levels) are available on the forms. Here is an example scenario to demonstrate the erroneous input. Let's say the user wants to view the graph of NOGAPS for `air_temp` at the surface level. The user clicks `air_temp` for the parameter, 0 for the level (the correct surface level for `air_temp` is 2.0 meters), `raob_qc` for the observation type (`raob_qc` is not a surface level observation type). The correct selections would have been `air_temp`, 2 meters, and `sfc_land` or `sfc_ship_met_qc`. The users for whom this research is designed are familiar with these facts. However, this is a definite shortcoming when this model verification system's user group expands in the future.

6. Other Open Issues

The current hardware configuration in FNMOC has a severe impact on the response time in the model verification system. The intranet web server is on an older and slower workstation (devu1) while the model statistics database is on a different workstation (div60-3). This hardware configuration requires a communication (handshake) from devu1 to div60-3. The model verification system then has to use the `runjob` script to query the database and create a GIF image on div60-3 from devu1. Once the GIF image is created, another handshake from div60-3 to devu1 is needed to send the GIF image back to devu1 from div60-3 by FTP. The first handshake takes approximately one minute while the actual data query and image generations take approximately seven seconds. It takes another 1 to 2 seconds to transfer the GIF file. The delays associated with the handshakes, `runjob` and `ftp` could be eliminated if the database and web server were on a single, faster workstation. The specific hardware requirements for a single server for web, database, and graphics need to be identified. Having hardware with the appropriate capability would also ease the development and testing process. It would eliminate the need for various routines such as `runjob` and `ftp`. This would decrease the time needed to process the user request by eliminating the need for the various hardware systems to communicate with each other. Finally, this would also help in planning for the future installation of the model verification system on the internet for external users.

LIST OF REFERENCES

1. Yourdon, Edward, "Modern Structured Analysis," Yourdon Press, 1989.
2. Kaner, Cem and Falk, Jack and Nguyen, Hung Quoc, "Testing Computer Software," 2nd Edition, International Thomson Computer Press, 1993.
3. Elmasri, Ramez and Navathe, Shamkant B., "Fundamentals of Database Systems," 2nd Edition, The Benjamin/Cummings Publishing Company, Inc., 1994.
4. Pressman, Roger S., "Software Engineering, A Practitioner's Approach," 3rd Edition, McGraw Hill Inc., 1992.
5. "Empress SQL User's Guide," Version 4, Empress Software Incorporated, 1990.
6. "Empress SQL Guide," Version 4, Empress Software Incorporated, 1990.
7. "IDL Reference Guide," Version 3.5, Research Systems, Inc. November 1993.
8. "IDL User's Guide," Version 3.5, Research Systems, Inc., November 1993.
9. Musciano, Chuck and Kennedy, Bill, "HTML The Definitive Guide," O'Reilly & Associates, Inc., 1996.
10. Haltiner, George J. and Williams, Roger Terry, "Numerical Prediction and Dynamic Meteorology," 2nd Edition, John Wiley and Sons, Inc., 1980.
11. Lines, Stephen, "How to Program CGI with Perl 5.0," Macmillan Computer Publishing USA, 1996.
12. Hoke, James E. and Hayes, John L. and Renninger, Larry G., "Map Projections and Grid Systems for Meteorological Applications," United States Air Force Air Weather Service, Air Force Global Weather Central, Offutt AFC NE 68113, March 1985.
13. "FNMOC Quarterly Performance Summary," summer 1997.
14. Rosenberg, Barry, "KornShell Programming Tutorial," Addison-Wesley Publishing Company, 1991.
15. Doswell, Charles A., "Forecaster Workstation Design: Concepts and Issues," NOAA/ERL National Wevere Storms Laboratory, Norman, Oklahoma, January 1992.
16. Mahoney, Jennifer Luppens, Henderson, Judy K., Miller, Patricia A., "A Description of the Forecast Systems Laboratory's Real-Time Verification System (RTVS)," NOAA Forecast Systems Laboratory, Boulder, CO, November, 1996.
17. Bernstein, L., Appel, J. J., "Requirements or Prototyping? Yes!," AT&T Bell Laboratories, USA.
18. Luqi, Shing, Mantak, "CAPS-A Tool For Real-Time System Development and Acquisition," Naval Research Review, 1992.
19. Williams, Neil, "The Benefits and Limitations of Software Prototyping," *Software Prototyping*, February 1995.
<http://osiris.sund.ac.uk/~calnwi/html/swproto/swpintro.html>

APPENDIX A

A. SMS VARIABLES

The following are common SMS variables for all the models:

```
ISIS_INIT=/a/ops/isis/db_init/init_ops.ksh  
OPSPATH=/a/ops  
OPSBIN=/a/ops/bin  
CRDATE=$(dtg)
```

1. NOGAPS

```
TAUI=0  
TAUE=144  
TAUINC=12  
GEOMNM1=global_360x181  
GEOMNM2=global_360x181  
ISIS_TABLE=NOGAPS  
VERIF_SOURCE=obs
```

2. NORAPS_ASIA

```
TAUI=0  
TAUE=48  
TAUINC=12  
GEOMNM1=asia_nest1_appl  
GEOMNM2=asia_nest1_appl  
ISIS_TABLE=NORAPS_ASIA  
VERIF_SOURCE=obs
```

3. NORAPS_CONUS

```
TAUI=0  
TAUE=48  
TAUINC=12  
GEOMNM1=conus_nest1_appl  
GEOMNM2=conus_nest1_appl  
ISIS_TABLE=NORAPS_CONUS  
VERIF_SOURCE=obs
```

4. NORAPS_EUROPE

TAUI=0
TAUE=48
TAUINC=12
GEOMNM1=europe_nest1_appl
GEOMNM2=europe_nest1_appl
ISIS_TABLE=NORAPS_EUROPE
VERIF_SOURCE=obs

5. NORAPS_IND_OCN

TAUI=0
TAUE=48
TAUINC=12
GEOMNM1=ind_ocn_nest1_appl
GEOMNM2=ind_ocn_nest1_appl
ISIS_TABLE=NORAPS_IND_OCN
VERIF_SOURCE=obs

6. NOGAPS FOR ASIA

TAUI=0
TAUE=48
TAUINC=12
GEOMNM1=global_360x181
GEOMNM2=asia_nest1_appl
ISIS_TABLE=NOGAPS
VERIF_SOURCE=obs

7. NOGAPS FOR CONUS

TAUI=0
TAUE=48
TAUINC=12
GEOMNM1=global_360x181
GEOMNM2=conus_nest1_appl
ISIS_TABLE=NOGAPS
VERIF_SOURCE=obs

8. NOGAPS FOR EUROPE

TAUI=0

TAUE=48
TAUINC=12
GEOMNM1=global_360x181
GEOMNM2=europe_nest1_appl
ISIS_TABLE=NOGAPS
VERIF_SOURCE=obs

9. NOGAPS FOR EUROPE_NEST2

TAUI=0
TAUE=48
TAUINC=12
GEOMNM1=global_360x181
GEOMNM2=europe_nest2_appl2
ISIS_TABLE=NOGAPS
VERIF_SOURCE=obs

10. NOGAPS FOR EUROPE_NEST3

TAUI=0
TAUE=24
TAUINC=6
GEOMNM1=global_360x181
GEOMNM2=europe_nest3_appl3
ISIS_TABLE=NOGAPS
VERIF_SOURCE=obs

11. COAMPS_EUROPE

TAUI=0
TAUE=48
TAUINC=12
GEOMNM1=europe_nest2_appl2
GEOMNM2=europe_nest2_appl2
ISIS_TABLE=COAMPS_EUROPE
VERIF_SOURCE=obs

12. COAMPS_EUROPE FOR NEST3

TAUI=0
TAUE=24
TAUINC=6
GEOMNM1=europe_nest3_appl3

GEOMNM2=europe_nest3_appl3
ISIS_TABLE=COAMPS_EUROPE
VERIF_SOURCE=obs

13. COAMPS_SOUTHWEST_ASIA FOR NEST2

TAUI=0
TAUE=48
TAUINC=12
GEOMNM1=southwest_asia_nest2_appl
GEOMNM2=southwest_asia_nest2_appl
ISIS_TABLE=COAMPS_SOUTHWEST_ASIA
VERIF_SOURCE=obs

14. COAMPS_SOUTHWEST_ASIA FOR NEST3

TAUI=0
TAUE=24
TAUINC=6
GEOMNM1=southwest_asia_nest3_appl
GEOMNM2=southwest_asia_nest3_appl
ISIS_TABLE=COAMPS_SOUTHWEST_ASIA
VERIF_SOURCE=obs

15. WAM_GLOBAL

TAUI=0
TAUE=144
TAUINC=12
GEOMNM1=global_360x181
GEOMNM2=global_360x181
ISIS_TABLE=WAM_GLOBAL
VERIF_SOURCE=obs

B. NAMELIST REQUIREMENTS OBTAINED FROM THE USERS FOR EACH MODEL

Table 3. Namelist information obtained from the users for each model

WAM_GLOBAL										
parm	typlvl	units	lvl_1	dsetname	llt parm	llt dsetname	llt seq_type	qc flag	obs time window	llt header file
sig_wav_ht	surface	m	0	fcst_ops	inst_wav_ht_2	decoded	sfc_ship	none	+/-12	SFC_SHIP.H
sig_wav_ht	surface	m	0	fcst_ops	sig_wav_ht	satdat	alty	none	+/-12	ALTY.H
peak_wav_per	surface	s	0	fcst_ops	inst_wav_per	decoded	sfc_ship		+/-12	SFC_SHIP.H

PIPS N_HEM										
parm	typ vl	units	lvl_1	dsetname	llt parm	llt dsetname	llt seq_type	qc flag	obs time window	llt header file
ice_cvr	surface	fraction	0	fcst_ops	ice_conc	decoded	ssmi_edr_ice	none	+/-12	SSMI_ED_I_ICE
sea_temp	surface	deg_K	0	fcst_ops	sea_temp	decoded	mcsst	sea_temp_qc_id	+/-12	MCSST.H

OTIS GLOBAL										
parm	typlvl	units	lvl_1	dsetname	llt parm	llt dsetname	llt seq_type	qc flag	obs time window	llt header file
sea_temp	surface	deg_K	0	anal_ops	sea_temp	decoded	mcsst	sea_temp_qc_id	+/-12	MCSST.H
sea_temp	surface	deg_K	0	anal_ops	sea_temp	metoc_qc	buoy	pos_qc_id, sea_temp_qc_id	+/-12	BUOY.H
sea_temp	surface	deg_K	0	anal_ops	sea_temp	metoc_qc	sfc_ship	pos_qc_id, sea_temp_qc_id	+/-12	SFC_SHI_P.H
sea_temp	surface	deg_K	0	anal_ops	sea_temp	metoc_qc	bthy	pos_qc_id, sea_temp_qc_id	+/-12	BTHY.H

OTIS W ATL										
parm	typlvl	units	lvl_1	dsetname	llt parm	llt dsetname	llt seq_type	qc flag	obs time window	llt header file
sea_temp	dpth_sfc	deg_K	0	anal_ops	sea_temp	metoc_qc	bthy		+/-12	BTHY.H

OTIS W PAC										
parm	typ vl	units	lvl_1	dsetname	llt parm	llt dsetname	llt seq_type	qc flag	obs time window	llt header file
sea_temp	dpth_sfc	deg_K	0	anal_ops	sea_temp	metoc_q c	bthy	pos_qc_id, sea_temp_ qc_id	+/-12	BTHY.H

TOPS GLOBAL										
parm	typ vl	units	lvl_1	dsetname	llt parm	llt dsetname	llt seq_type	qc flag	obs time window	llt header file
sea_temp	dpth_sfc	deg_K	0	fcst_ops	sea_temp	metoc_q c	bthy		+/-12	BTHY.H

TOPS W ATL										
parm	typ vl	units	lvl_1	dsetname	llt parm	llt dsetname	llt seq_type	qc flag	obs time window	llt header file
sea_temp	dpth_sfc	deg_K	0	fcst_ops	sea_temp	metoc_q c	bthy		+/-12	BTHY.H

TOPS W PAC										
parm	typlvl	units	lvl_1	dsetname	llt parm	llt dsetname	llt seq_type	qc flag	obs time window	llt header file
sea_temp	dpth_sfc	deg_K	0	fcst_ops	sea_temp	metoc_qc	bthy		+/-12	BTHY.H

NOGAPS, ALL NORAPS, ALL COAMPS										
parm	typlvl	units	lvl_1	dsetname	llt parm	llt dsetname	llt seq_type	qc flag	obs time window	llt header file
air_temp	isbr_lvl	deg_K	*	fcst_ops	air_temp	fmoc	raob_qc	air_temp_qc_id		RAOB_QC.H
air_temp	ht_sfc	deg_K	2.0	fcst_ops	air_temp	fmoc	sfc_ship_met_qc	air_temp_qc_id		SFC_SH_IP_MET_QC.H
air_temp	ht_sfc	deg_K	2.0	fcst_ops	air_temp	fmoc	sfc_lnd	air_temp_qc_id		SFC_LN_D.H
geop_ht	isbr_lvl	gpm	*	fcst_ops	geop_ht	fmoc	raob_qc	geop_ht_qc_id		RAOB_QC.H
pres	msl	mb	0.0	fcst_ops	sea_lvl_pres	fmoc	sfc_ship_met_qc	sea_lvl_pres_qc_id		SFC_SH_IP_MET_QC.H
pres	msl	mb	0.0	fcst_ops	sea_lvl_pres	fmoc	sfc_lnd	sea_lvl_pres_qc_id		SFC_LN_D.H

NOGAPS, ALL NORAPS, ALL COAMPS									
wnd_spd	isbr_lvl	m/s	*	fcst_ops	wnd_spd	fmoc	raob_qc	wnd_qc_id	RAOB_QC.H
wnd_spd	ht_sfc	m/s	19.5	fcst_ops	wnd_spd	fmoc	sfc_lnd	wnd_qc_id	SFC_LN D.H
wnd_spd	ht_sfc	m/s	19.5	fcst_ops	wnd_spd	fmoc	sfc_ship_met_qc	wnd_qc_id	SFC_SH IP_MET QC.H

* 1000, 925, 850, 700, 500, 400 300, 250, 200, 150, 100

C. NON-SURFACE NAMELIST FILES

1. NOGAPS, all NORAPS and all COAMPS

```
&verlst
parm="air_temp", "geop_ht", "wnd_spd",
dsetname="fcst_ops", "fcst_ops", "fcst_ops",
obs_parm="air_temp", "geop_ht", "wnd_spd",
odsetname="fnmoc", "fnmoc", "fnmoc",
units="deg_K", "gpm", "m/s",
lvltype="isbr_lvl", "isbr_lvl", "isbr_lvl",
obstype="raob_qc", "raob_qc", "raob_qc",
level=1000,925,850,700,500,400,300,250,200,150,100,
stats="bias", "rms", "std",
tval=35.0
&end
```

D. SURFACE NAMELIST FILES

1. NOGAPS, all NORAPS and all COAMPS

```
&sfcslst
sfc_parm="air_temp", "air_temp", "pres", "pres", "wnd_spd", "wnd_spd",
sfc_dsetname="fcst_ops", "fcst_ops", "fcst_ops", "fcst_ops", "fcst_ops", "fcst_ops",
sfc_obs_parm="air_temp", "air_temp", "sea_lvl_pres", "sea_lvl_pres", "wnd_spd", "wnd_spd",
sfc_odsetname="fnmoc", "fnmoc", "fnmoc", "fnmoc", "fnmoc", "fnmoc",
sfc_units="deg_K", "deg_K", "mb", "mb", "m/s", "m/s",
sfc_lvltype="ht_sfc", "ht_sfc", "msl", "msl", "ht_sfc", "ht_sfc",
sfc_obstype="sfc_lnd", "sfc_ship_met_qc", "sfc_lnd", "sfc_ship_met_qc", "sfc_lnd", "sfc_ship_met_qc",
sfc_level=2.0,2.0,0.0,0.0, 19.5,19.5,
sfc_stats="bias", "rms", "std",
sfc_tval=35.0
&end
```

2. WAM_GLOBAL

```
&verlst  
parm="sig_wav_ht", "peak_wav_per",  
obs_parm="inst_wav_ht_2", "inst_wav_per",  
units="m", "s",  
lvltype="surface", "surface",  
level=0,0,  
stats="bias", "rms", "std",  
obstype="sfc_ship",  
tval=35.0  
&end
```

APPENDIX B SOURCE CODE

A. OPS RUN

1. mverif.job

```
# QSUB -lt 00:30:00
# QSUB -IT 00:30:00
# QSUB -lm 10Mw
# QSUB -lM 10MW
# QSUB -eo
# QSUB -ko
# QSUB -o /home/pacek/mverif/etc/mverif.out
# QSUB -s /bin/ksh
# QSUB -x

=====
# local function to handle the exit code from executing binary
=====
INIT_user_exit() {

case $? in
0) #Normal exit
  INIT_joblog_comment=MVERIF_COMPLETED_OK
  code=0
  ;;
1) #pfxgetenv error in the main program
  INIT_joblog_comment=MVERIF_INFORMATIVE_PFXGETENV_ERROR
  code=0
  ;;
2) #ch2int error in the main program
  INIT_joblog_comment=MVERIF_INFORMATIVE_CH2INT_ERROR
  code=0
  ;;
3) #not enough info to continue
  INIT_joblog_comment=MVERIF_INFORMATIVE_NOT_ENOUGH_INFO
  code=0
  ;;
4) #output file open error
  INIT_joblog_comment=MVERIF_INFORMATIVE_OUTPUT_FILE_OPEN_ERR
  code=0
  ;;
5) #error with ggrd call
  INIT_joblog_comment=MVERIF_INFORMATIVE_GGRD_CALL_ERR
  code=0
  ;;
6) #error with getgeom call
```

```

INIT_joblog_comment=MVERIF_INFORMATIVE_GETGEOM_CALL_ERR
code=0
;;
7) #error with boundary subroutine call
INIT_joblog_comment=MVERIF_INFORMATIVE_BOUNDARY_CALL_ERR
code=0
;;
*) #Abnormal exit, save log file
INIT_joblog_comment=MVERIF_FAILED
code=99
if [ -f "$tempfile" ]; then
    cp $tempfile $HOME/mverif/etc/o_mverif_err.log
fi
;;
esac
return $code
}

#=====
# setup the job environment
#=====
. $OPSBIN/init_job
set -Sx
INIT_do_not_notify_operator=0
INIT_notify_status=ERR
INIT_notify_users="pacek"
. $ISIS_INIT

#=====
# test to see if env var CRDATE exist, otherwise take the ops dtg
#=====
if [ -z "$CRDATE" ]
then
    echo "Null CRDATE, setting to ops dtg."
    CRDATE=$(dtg)
    export CRDATE
fi

#=====
# set the required env var and export the SMS and other var
#=====
#PROGBIN=/home/pacek/mverif/bin
PROGBIN=${PROGBIN:-$OPSBIN}
MVERIF_DIR=$OPSPATH/etc/dynamic/app/mverif
#MVERIF_DIR=$HOME/mverif/etc
echo $MVERIF_DIR
integer TAUI
integer TAUE
integer TAUINC
export TAUI TAUE TAUINC
#TEST_DIR=/home/pacek/mverif/test
TEST_DIR=$OPSPATH/etc/static/app/mverif

```

```

NAMLIST_FILE1=$ISIS_TABLE-verfil
NAMLIST_FILE2=$ISIS_TABLE-sfverfil
print "namlist_file1=" $NAMLIST_FILE1 " namlist_file2=" $NAMLIST_FILE2
export NAMLIST_FILE1 NAMLIST_FILE2

```

```

#=====
# job accounting info
#=====
ja

```

```

#=====
# need to pull out the month and year from CRDATE
#=====

```

```

year=`echo $CRDATE | cut -c1-4`
month=`echo $CRDATE | cut -c5-6`
case $month in
  01) mon='jan' ;;
  02) mon='feb' ;;
  03) mon='mar' ;;
  04) mon='apr' ;;
  05) mon='may' ;;
  06) mon='jun' ;;
  07) mon='jul' ;;
  08) mon='aug' ;;
  09) mon='sep' ;;
  10) mon='oct' ;;
  11) mon='nov' ;;
  12) mon='dec' ;;
esac

```

```

#=====
# find the appropriate subdirectory
#=====
subdir=$mon$year

```

```

#=====
# need to test for the existence of subdir otherwise the
# job will crash
#=====

```

```

if [ -d $MVERIF_DIR/$subdir ]
then
  echo $MVERIF_DIR/$subdir "exists"
else
  mkdir $MVERIF_DIR/$subdir
  chmod 775 $MVERIF_DIR/$subdir
fi

```

```

#=====
# the output directory
#=====

```

```

OUTDIR=$MVERIF_DIR/$subdir
echo $ISIS_TABLE $TAUI $TAUE $TAUINC

```

```

#=====
# determine the current watch and month's stat file name
#=====
curr_file=$ISIS_TABLE$GEOMNM1$CRDATE
month_file=$ISIS_TABLE$GEOMNM1$mon$year
if [[ $GEOMNM2 != " ]]
then
    curr_file=$ISIS_TABLE$GEOMNM2$CRDATE
    month_file=$ISIS_TABLE$GEOMNM2$mon$year
fi

#=====
# if curr_file exists (ran once) delete so program will not exit
#=====
if [ -f $OUTDIR/$curr_file ]
then
    echo "removing stat out file"
    rm $OUTDIR/$curr_file
fi

#=====
# to use $TMPDIR, the scratch area
#=====
cd $TMPDIR

#=====
# copy the namelist files to the scratch area
#=====
if [[ -f $TEST_DIR/$NAMLIST_FILE1 ]]
then
    cp $TEST_DIR/$NAMLIST_FILE1 $NAMLIST_FILE1
fi

if [[ -f $TEST_DIR/$NAMLIST_FILE2 ]]
then
    cp $TEST_DIR/$NAMLIST_FILE2 $NAMLIST_FILE2
fi

#=====
# execute mverif program
#=====
if [[ $VERIF_SOURCE = 'obs' ]]
then
    print "calling verobs"
    $PROGBIN/verobs
elif [[ $VERIF_SOURCE = 'anal' ]]
then
    print "calling veranal"
    $PROGBIN/veranal
else
    print "VERIF_SOURCE must be either 'obs' or 'anal'."
    exit 8

```

fi

```
#=====
# append the current run's stat to the monthly stat file
#=====
#cp $curr_file $OUTDIR/$curr_file
cat $curr_file >> $OUTDIR/$month_file

#=====
# close up the job accounting info and clean up
#=====
rm -f core
ja -st

#.....START EPILOGUE.....
#
# SCCS IDENTIFICATION: @(#)mverif.job 1.1 04/24/98 /h/cm/library/mverif/src/job/mverif.job_v
#
# CONFIGURATION IDENTIFICATION:
#
# SCRIPT NAME: mverif.job
#
# SHELL TYPE: Korn
#
# DESCRIPTION: Script that runs MVERIF which computes verifying stats of
#              the models against the LLT observations or VERANAL which
#              computes verifying stats against the model analysis.
#
# COPYRIGHT:   (c) 1998 FLENUMMETOCEN
#              U.S. GOVERNMENT DOMAIN
#              ALL RIGHTS RESERVED
#
# CONTRACT NUMBER AND TITLE: NONE
#
# REFERENCES: NONE
#
# CLASSIFICATION: Unclassified
#
# RESTRICTIONS: NONE
#
# COMPUTER/OPERATING SYSTEM   Cray UNICOS
# DEPENDENCIES:
#
# LIBRARIES OF RESIDENCE: /a/ops/app/mverif/src/job
#
# USAGE: qsub mverif.job
#
# PARAMETERS: SMS variables needed
#   Name           Description
#   -----
#   ISIS_INIT      ISIS init script
#   CRDATE         current run dtg
```

```

# OPSPATH      ops path
# OPSBIN       binary directory
# PROGBIN      test binary directory
# MVERIF_DIR   mverif stat output files directory
# GEOMNM1      geometry name (e.g., conus_nest1_appl)
# GEOMNM2      geometry name to interpolate to
# ISIS_TABLE   model name (e.g., NORAPS_CONUS)
# TAU1         starting tau
# TAU2         finishing tau
# TAUINC       tau increment
#
# RETURN CODE:
#
# FILES:
#   Name      Usage      Description
#   -----
#   curr_file  IN         file that contains the current run's stats
#   month_file IN/OUT     file that contains the month's stats
#
# DATA BASES:
#   Name      Table      Usage      Description
#   -----
#
# NON-FILE INPUT/OUTPUT:
#   Name      Type      Usage      Description
#   -----
#
# ERROR CONDITIONS:
#   Condition      Action
#   -----
#   data not found      stop executing
#   curr_file not found  cannot append to month_file,
#                       sends an email to group
#
# ADDITIONAL COMMENTS: NONE
#
#.....MAINTENANCE SECTION.....
#
# EXTERNALS CALLED:
#   Name      Description
#   -----
#   verobs    reads ISIS grid data, ISIS LLT data and computes
#             the models' verifying stats
#   veranal   reads model forecast and analysis from ISIS grid
#             and computes the models' verifying stats
#
# VARIABLES:
#   Name      Description
#   -----
#   year      4-digit year from $CRDATE
#   month     2-digit month from $CRDATE
#   mon       3-character month (e.g., jan)
#   NAMELIST_FILE1  namelist file name for upper levels

```

```
# NAMELIST_FILE2    namelist file name for surface level
#
# METHOD: change directory to the $TMPDIR and run the Fortran program
#   verobs or veranal, copy the current run's stat file to the
#   $OPSPATH and append the current run's stat file to the
#   monthly stat file in the $OPSPATH
#
# RECORD OF CHANGES:
# <<CHANGE NOTICE>> version 1.1 (29 Apr 1998) -- Kyongsuk Pace
#   initial submission
#
#.....END EPILOGUE.....
```

2. statupd.ksh

```
#!/bin/ksh
#=====
# inserts the model stats into the stat_db
#=====
DATA_DIR=/home/pacek/data
DB_DIR=/d/model-stats
cd $DB_DIR

#=====
# field separation for Empress
#=====
export MSValsep=' '

#=====
# is there a new data file? if so, move them
#=====
if [[ -a $DATA_DIR ]]
then
  mv $DATA_DIR/* .
else
  print "There are no data files!"
fi

#=====
# determine which model by looking at the files that
# ends with numeric 0 or 2
# example: NOGAPSGlobal_360x1811998021000
#=====
for OBJ in *[0-9]
do
  print $OBJ
  awk '/^[0-9]+/ { print $1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12 }' \
  $OBJ > tmp

  case $OBJ in
    NOGAPSasia_nest1_appl*)
```

```

    empcmd stat_db "insert into nogaps_asia_nest1_appl from tmp";,;
NOGAPSconus_nest1_appl*)
    empcmd stat_db "insert into nogaps_conus_nest1_appl from tmp";,;
NOGAPSeurope_nest1_appl*)
    empcmd stat_db "insert into nogaps_europe_nest1_appl from tmp";,;
NOGAPSeurope_nest2_appl2*)
    empcmd stat_db "insert into nogaps_europe_nest2_appl2 from tmp";,;
NOGAPSeurope_nest3_appl3*)
    empcmd stat_db "insert into nogaps_europe_nest3_appl3 from tmp";,;
NOGAPSGlobal*)
    empcmd stat_db "insert into nogaps_global_360x181 from tmp";,;
NOGAPScind_ocn_nest1_appl*)
    empcmd stat_db "insert into nogaps_ind_ocn_nest1_appl from tmp";,;
NORAPS_ASIA*)
    empcmd stat_db "insert into noraps_asia from tmp";,;
NORAPS_CONUS*)
    empcmd stat_db "insert into noraps_conus from tmp";,;
NORAPS_EUROPE*)
    empcmd stat_db "insert into noraps_europe from tmp";,;
NORAPS_IND_OCN*)
    empcmd stat_db "insert into noraps_ind_ocn from tmp";,;

COAMPS_SOUTHWEST_ASIA*)
    empcmd stat_db "insert into coamps_sw_asia from tmp";,;
COAMPS_EUROPE*)
    empcmd stat_db "insert into coamps_europe from tmp";,;
WAM_GLOBAL*)
    empcmd stat_db "insert into wam_global from tmp";,;
*)
    print "no stat table for " $OBJ;
esac

mv $OBJ /home/pacek/backup
done

```

3. v_data.h

```

!.....START PROLOGUE.....
!
! SCCS IDENTIFICATION: @(#)v_data.h 1.1 04/24/98
!
! RECORD OF CHANGES:
! <<CHANGE NOTICE>> Version 1.1 (29 Apr 1998) -- Kyongsuk Pace
!   Initial submission
!
!.....END PROLOGUE.....

!*****
!The Fortran include file V_DATA will hold all the data definitions
!used throughout the verobs and its subroutines. This will
!help in modifying at one point if the requirement happens to

```

!change in the future.

!*****

```
integer :: im
integer :: jm
integer :: ijmax ! 1-dim array max size
integer :: maxobs ! max num of obs
integer :: maxprm ! max num of parameters to verify
integer :: maxstat ! max num of stats to verify
integer :: maxtv1 ! max num of threshold values
integer :: maxlvl ! max num of levels
integer :: size
real :: bad_value ! value for missing or bad data
real :: check_val ! ISIS missing value checking number
```

```
parameter(im=360)
parameter(jm=181)
parameter(ijmax=im*jm)
parameter(maxobs=50000)
parameter(maxprm=20)
parameter(maxstat=10)
parameter(maxtv1=20)
parameter(maxlvl=30)
parameter(size=5000)
parameter(bad_value=1.E+10)
parameter(check_val=1.E+9)
```

!-----

! env var

!-----

```
character(16) :: crdate_val ! current run dtg value
character(8) :: tau1_val ! starting tau value
character(8) :: tau2_val ! ending tau value
character(8) :: tauinc_val ! tau increment value
character(32) :: modelname ! model name value
character(32) :: geomname ! geometry name value
character(24) :: prjnm ! projection name
character(16) :: vdtg ! verifying dtg, also for write
integer :: ngeom ! C pointer for the given geomname
integer :: ncols ! number of columns
integer :: nrow ! number of rows
integer :: itau1 ! integer beginning tau
integer :: itau2 ! integer ending tau
integer :: itauinc ! integer tau increment
```

4. verobs.f90

program verobs

C

C.....START PROLOGUE.....

C

CSCCS IDENTIFICATION: @(#)verobs.f901.1 04/24/98 /h/cm/library/mverif/src/main/verobs.f90_v

C
 C CONFIGURATION IDENTIFICATION: NONE
 C
 C MODULE NAME: verobs
 C
 C DESCRIPTION: This program verifies the model forecast fields against
 C observations for various models.
 C
 C COPYRIGHT: (c) 1998 FLENUMMETOCEN
 C U.S. GOVERNMENT DOMAIN
 C ALL RIGHTS RESERVED
 C
 C CONTRACT NUMBER AND TITLE: N/A
 C
 C REFERENCES: previous verobs.f
 C
 C CLASSIFICATION: Unclassified
 C
 C RESTRICTIONS: NONE
 C
 C COMPUTER/OPERATING SYSTEM
 C DEPENDENCIES: Cray UNICOS
 C
 C LIBRARIES OF RESIDENCE: /a/ops/bin
 C
 C USAGE: Korn shell script mverif.job
 C
 C PARAMETERS: N/A
 C
 C COMMON BLOCKS: N/A
 C
 C FILES:

Name	Unit	Type	Attribute	Usage	Description
\$MODEL-verfil	10	FORMATTED DIRECT	IN	Contains parameter, stat types to compute	
\$MODEL-sfcverfil	10	FORMATTED DIRECT	IN	Contains parameter, stat types to compute for sfc	
MODELGEOMDTG	10	FORMATTED DIRECT	OUT	Contains the computed stats (e.g., NORAPSconus_nest1_appl1996010100)	for each run

Name	Table	Usage	Description
ISIS Grid data	Various	IN	Model Forecasts
ISIS LLT data	Various	IN	Observed environmental data

 C NON-FILE INPUT/OUTPUT: N/A
 C
 C ERROR CONDITIONS:

CONDITION	ACTION
-----	-----

```

C   no env variables      exit with code 1
C   err in char to integer
C   conversion            exit with code 2
C   empty input arrays    exit with code 3
C   err opening output file exit with code 4
C   ggrd error            exit with code 5
C   getgeom error         exit with code 6
C   boundary error        exit with code 7
C   no ISIS Grid data     stop executing
C   no ISIS LLT data      stop executing
C
C ADDITIONAL COMMENTS: NONE
C
C.....MAINTENANCE SECTION.....
C
C MODULES CALLED:
C   Name      Description
C   -----
C   BOUNDARY   Computes min/max lat/lon for reading obs from LLT DB
C   CH2INT     Converts character to integer
C   DBSTOP     ISIS software that terminates database
C   DTGMOD     FNOC utility that increments a DTG
C   EXIT       System call that exits program
C   LLTREAD    Reads obs from LLT DB
C   PXFGETENV  Gets environment variable
C   GETGEOM    Gets geometry arguments to be used by other routines
C   GGRD       Returns ISIS info. on given geometry
C   GRD        Reads gridded fcst fields from ISIS
C   STRLEN     FORTRAN function that returns string length
C   F2OB       Interpolates fcst to obs pts
C   UV2DF      Converts wind u/v to direction and speed
C
C LOCAL VARIABLES AND   Structures are documented in detail
C   STRUCTURES:         where they are defined in the code
C                       within include files.
C
C INCLUDE FILES:
C   Name      Description
C   -----
C
C COMPILER DEPENDENCIES: empef90
C
C COMPILE OPTIONS: -f fixed -c
C
C MAKEFILE: Located at /a/ops/app/mverif/src/main/Makefile
C   UNICOS make
C
C RECORD OF CHANGES:
C <<CHANGE NOTICE>> Version 1.1 (29 Apr 1998) -- Kyongsuk Pace
C   Initial submission
C
C.....END PROLOGUE.....

```

```

C
  implicit none
  include 'v_data.h'

c*****
c  Local variables
c*****
  ! record type
  TYPE ver_data
    character(32) :: param
    character(24) :: dsetnm
    character(32) :: obs_param
    character(24) :: odsetnm
    character(32) :: unit
    character(24) :: typlvl
    real          :: lvl_1
    character(24) :: obs_type
  END TYPE ver_data

c*****
  !array of records
c*****
  TYPE(ver_data) verif(maxprm*maxlvl),
  2          sfc_verif(maxprm)

  character(1) :: cnul !one blank space var used to initialize
  character(8) :: seclvl !secret level returned from GRD
  character(10):: dtg !10 char long crdate_val
  character(16):: cdtg !dtgmod applied dtg, i.e. dtg-0,12,24, etc.
  character(16):: tdtg !temp cdtg
  character(16):: stats(maxstat), sfc_stats(maxstat) !stat types
  character(24):: stdesc !storage description returned from GETGEOM
  character(24):: lvltype(maxprm), sfc_lvltype(maxprm)!level types
  character(24):: obstype(maxprm), sfc_obstype(maxprm)
    !l1 seq type (raob_qc, sfc_ship, etc.)
  character(24):: dsetname(maxprm), sfc_dsetname(maxprm)
  character(24):: odsetname(maxprm), sfc_odsetname(maxprm)
  character(32):: parm(maxprm), sfc_parm(maxprm) !parameters
  character(32):: obs_parm(maxprm), sfc_obs_parm(maxprm)
  character(32):: units(maxprm), sfc_units(maxprm)
  character(32):: geomname2, outgeomname
  character(60):: outstats !output filename
  character(40):: namlist_file1_val, namlist_file2_val
  character(80):: title
  character(4)  :: nul_geom

  integer :: nstat, nparm, nobs, nobstype, nlevel
  integer :: sfc_nstat, sfc_nparm, sfc_nobs, sfc_nobstype
  integer :: arr_size, sfc_arr_size

  integer :: ktau, ltau, lstat, i, j, k, l, m, n
  integer :: numchar, istat

```

```

integer    :: id, iseq, status

integer    :: lenMODEL, lenmodelname
integer    :: lenGEOMNM, lengeomname
integer    :: lenGEOMNM2, lengeomname2
integer    :: lenTAUI, lentau_val
integer    :: lenTAUE, lentaue_val
integer    :: lenTAUINC, lentauinc_val
integer    :: lenCRDATE, lencrdate_val
integer    :: lenNAMLIST_FILE1, lennamlist1_val
integer    :: lenNAMLIST_FILE2, lennamlist2_val

real       :: fcst(ijmax), fcstu(ijmax), fcstv(ijmax)
real       :: fcst2(ijmax), fcstu2(ijmax), fcstv2(ijmax)
real       :: fu(ijmax), fv(ijmax), fdir(maxobs)
real       :: oblat(maxobs), oblon(maxobs), obval(maxobs)
real       :: newlat(maxobs), newlon(maxobs), newobs(maxobs)
real       :: fob(maxobs)
real       :: newfob(maxobs)
real       :: newfu(ijmax), newfv(ijmax)
real       :: level(maxlvl), sfc_level(maxlvl)
real       :: origx, origy, parm1, parm2, parm3, xintdis,
2          yintdis, minlat, maxlat, minlon, maxlon
real       :: xlvl, level_2, paknul
real       :: tval, sfc_tval
real       :: bias, rms, std, ancor
real       :: ftau !float tau

```

```

double precision :: origlat, origlon

```

```

c*****

```

```

c  Function

```

```

c*****

```

```

integer strlen

```

```

c*****

```

```

c  Data Initialization

```

```

c*****

```

```

data cnul  /' '/
data lstat /10/
data paknul /10.e10/
data level_2 /0.0/

```

```

c*****

```

```

c  Namelists

```

```

c*****

```

```

namelist /verlst/ parm, dsetname, obs_parm, odsetname,
2  units, lvltype, obstype, level, stats, tval

```

```

namelist /sfcfst/ sfc_parm, sfc_dsetname, sfc_obs_parm,
2  sfc_odsetname, sfc_units, sfc_lvltype,

```

2 sfc_obstype, sfc_level, sfc_stats, sfc_tval

```
c*****
c initialize some var
c*****
  crdate_val(1:16) = cnul
  dtg(1:10) = cnul
  cdtg(1:16) = cnul
  seclvl = 'UNCLASS'

c*****
c Get the environment variables that are set in the job script,
c CRDATE, TAUS, TAUE, TAUINC, MODEL, GEOMNM, DATASET1, DATASET2,
c NAMLIST_FILE1 and NAMLIST_FILE2.
c*****
  namlist_file1_val = cnul
  namlist_file2_val = cnul
  geomname2 = cnul

  CALL PXFGETENV('ISIS_TABLE', lenMODEL, modelname, lenmodelname,
2      istat)
  if (istat .ne. 0) then
    write *, "MODEL is unspecified"
    CALL EXIT(1)
  end if

  CALL PXFGETENV('GEOMNM1', lenGEOMNM, geomname, lengeomname,
2      istat)
  if (istat /= 0) then
    write *, "GEOMNM1 is unspecified"
    CALL EXIT(1)
  end if

  CALL PXFGETENV('GEOMNM2', lenGEOMNM2, geomname2, lengeomname2,
2      istat)
  if (istat /= 0) then
    write *, "No second GEOMNM is specified"
  end if

  CALL PXFGETENV('TAUI', lenTAUI, tau_val, lentau_val, istat)
  if (istat /= 0) then
    write *, "TAUI is unspecified"
    CALL EXIT(1)
  endif

  CALL PXFGETENV('TAUE', lenTAUE, taue_val, lentaue_val, istat)
  if (istat /= 0) then
    write *, "TAUE is unspecified"
    CALL EXIT(1)
  endif

  CALL PXFGETENV('TAUINC', lenTAUINC, tauinc_val, lentauinc_val,
```

```

2      istat)
if (istat /= 0) then
  write *, "TAUINC is unspecified"
  CALL EXIT(1)
endif

CALL PXFGETENV('CRDATE', lenCRDATE, crdate_val, lencrdate_val,
2      istat)
if (istat /= 0) then
  write *, "CRDATE is unspecified"
  CALL EXIT(1)
endif

CALL PXFGETENV('NAMLIST_FILE1', lenNAMLIST_FILE1,
2      namlist_file1_val, lennamlist1_val, istat)
if (istat /= 0) then
  write *, "NAMLIST_FILE1 is unspecified"
endif

CALL PXFGETENV('NAMLIST_FILE2', lenNAMLIST_FILE2,
2      namlist_file2_val, lennamlist2_val, status)
if (status /= 0) then
  write *, "no second namlist file."
endif

if (istat /= 0 .and. status /= 0) then
  write *, "cannot continue without the namlist files."
  CALL EXIT(1)
endif

c*****
c  get ISIS info on the given geometry by calling GGRD
c  geom is defined as a SMS env var
c  will add the capability to add the geom info in the future
c  when the geom info is not defined in ISIS
c*****
CALL GGRD(geomname, ngeom, istat)
!if geom is not defined then exit the program
if (istat .ne. 0) then
  write (0, ("undefined geom, ggrd returns istat =", i5)) istat
  CALL EXIT(5)
endif

print *, 'taui=', taui_val, ' taue=', taue_val,
2      ' tauinc=', tauinc_val
print *, 'model=', modelname
print *, 'geomnm=', geomname

c*****
c  need to convert tau info to integers
c*****
CALL CH2INT(tau_val, itau, istat)

```

```

if (istat .ne. 0) then
  write (0,('ch2int on TAUJ returns istat =",i5')) istat
  CALL EXIT(2)
end if

CALL CH2INT(taue_val, itaue, istat)
if (istat .ne. 0) then
  write (0,('ch2int on TAUJ returns istat =",i5')) istat
  CALL EXIT(2)
end if

CALL CH2INT(tauinc_val, itauinc, istat)
if (istat .ne. 0) then
  write (0,('ch2int on TAUINC returns istat =",i5')) istat
  CALL EXIT(2)
end if

c*****
c  Initialize the arrays
c*****
do m=1, maxprm
  parm(m) = cnul
  dsetname(m) = cnul
  odsetname(m) = cnul
  units(m) = cnul
  lvltype(m) = cnul
  sfc_parm(m) = cnul
  sfc_dsetname(m) = cnul
  sfc_odsetname(m) = cnul
  sfc_units(m) = cnul
  sfc_lvltype(m) = cnul
  obstype(m) = cnul
  sfc_obstype(m) = cnul
end do

do m=1, maxlvl
  level(m) = paknul
  sfc_level(m) = paknul
end do

do n=1, maxstat
  stats(n) = cnul
  sfc_stats(n) = cnul
end do

c*****
!Open and read the NAMLIST_FILE1 into the arrays for parm,
!unit, lvl_type, level_1, and obs_type

!determine the number of parameters and number of levels
!number of stats and number of obstype
!determine the array size

```

```

c*****
  if (namlist_file1_val /= cnul) then
    open (unit=lstat,file=namlist_file1_val,form='formatted',
2      status='old',iostat=istat)
    if (istat .eq. 0) read(lstat, nml = verlst)

c*****
c  Finish setup, the arrays, stype, tval, typ1vl, units are changed
c  to 1-dimension and will not change throughout the program so that
c  they can be used for multiple parameters which is a change from
c  the original program.
c*****
c  Determine the number of parameters
c*****
  nparm = 0
  do while (parm(nparm+1) .ne. cnul .and. nparm .lt. maxprm)
    nparm = nparm + 1
  end do

  if (nparm .eq. 0) then
    write (0,('No verification parameters specified'))
c  CALL EXIT(3)
  end if

c*****
c  Determine the number of levels
c*****
  nlevel = 0
  do while (level(nlevel+1) .ne. paknul .and. nlevel .lt. maxlvl)
    nlevel = nlevel + 1
  end do

  if (nlevel .eq. 0) then
    write (0,('No verification levels specified.'))
c  CALL EXIT(3)
  end if

c*****
c  Determine the number of stats
c*****
  nstat = 0
  do while (stats(nstat+1) .ne. cnul .and.
2    nstat .lt. maxstat)
    nstat = nstat + 1
  end do

  if (nstat .eq. 0) then
    write (0,('No statistics are requested.'))
c  CALL EXIT(3)
  end if

  end if !namlist_file1 exists

```

```

c*****
c  if NAMLIST_FILE2 exists, read it and put into the arrays for
c  the surface
c*****
  if (namlist_file2_val /= cnul) then
    open (unit=lstat, file=namlist_file2_val,
2      form='formatted', status='old', iostat=istat)

    if (istat .eq. 0) then
      read(lstat, nml = sfc1st)
      sfc_nparm = 0
      do while (sfc_parm(sfc_nparm+1) .ne. cnul .and.
2        sfc_nparm .lt. maxprm)
        sfc_nparm = sfc_nparm + 1
      end do
      sfc_nstat = 0
      do while (sfc_stats(sfc_nstat+1) .ne. cnul .and.
2        sfc_nstat .lt. maxstat)
        sfc_nstat = sfc_nstat + 1
      end do
      if (sfc_nstat .eq. 0) then
        write (0, '("No statistics are requested in sfc.")')
c      CALL EXIT(3)
      end if
    end if !if namlist_file2 was read successfully
  end if !if namlist_file2 exists

c*****
c  array size based on the namlists
c*****
  arr_size = nparm*nlevel
  sfc_arr_size = sfc_nparm

c*****
!Fill the arrays of records from the parm, unit, lvl_type,
!level_1 and obs_type arrays
c*****
  l = 1
  if (arr_size .gt. 0) then
    do i = 1, nparm
      do j = 1, nlevel
        verif(l)%param = parm(i)
        verif(l)%dsetnm = dsetname(i)
        verif(l)%obs_param = obs_parm(i)
        verif(l)%odsetnm = odsetname(i)
        verif(l)%unit = units(i)
        verif(l)%typ1vl = lvltype(i)
        verif(l)%lvl_1 = level(j)
        verif(l)%obs_type = obstype(i)
        l = l + 1
      end do
    end do
  end do

```

```

end if

c*****
!add the sfc stuff to the verif array
c*****
if (sfc_arr_size .gt. 0) then
  arr_size = arr_size + sfc_arr_size
  do i = 1, sfc_nparm
    verif(1)%param = sfc_param(i)
    verif(1)%dsetnm = sfc_dsetname(i)
    verif(1)%obs_param = sfc_obs_param(i)
    verif(1)%odsetnm = sfc_odsetname(i)
    verif(1)%unit = sfc_units(i)
    verif(1)%typlvl = sfc_lvltypes(i)
    verif(1)%lvl_1 = sfc_level(i)
    verif(1)%obs_type = sfc_obstype(i)
    l = l + 1
  end do
end if

c*****
!get geom info
c*****
CALL GETGEOM(ngeom, prjnm, stdesc, ncols, nrow,
2      origlat, origlon, origx, origy, xintdis,
3      yintdis, parm1, parm2, parm3, istat)

if (istat .ne. 0) then
  write (0, '("getgeom returns istat =",i5)') istat
  CALL EXIT(6)
end if

c*****
c Determine min/max lat/lon to use in obs read
c*****
CALL BOUNDARY(ngeom, nrow, ncols, minlat,
2      maxlat, minlon, maxlon, istat)

if (istat .ne. 0) then
  write (0, '("cannot find min/max lat/lon")')
  CALL DBSTOP
  CALL EXIT(7)
end if

c*****
c The output file, use modelname, geomname and crdate_val
c*****
l = strlen(modelname)
outstats(1:l) = modelname(1:l)
if (geomname2 == cnul) then
  numchar = strlen(geomname)
  k = l + numchar

```

```

        outstats(l+1:k) = geomname(1:numchar)
    else
        numchar = strlen(geomname2)
        k = l + numchar
        outstats(l+1:k) = geomname2(1:numchar)
    end if

    outstats(k+1:k+10) = crdate_val
    write (0,('output file name = ', a50))
2    outstats(1:strlen(outstats))

    open(unit=lstat,file=outstats(1:strlen(outstats)),
2    form='formatted',status='new',iostat=istat)

    if (istat .ne. 0) then
        write (0,('iostat = ', i5)) istat
        write (0,('Cannot open output file for stats'))
        CALL EXIT(4)
    end if

c*****
c  Write the header in the output file
c*****
    write (lstat, ('  vdtg  ", " numobs ",
2        " param  ", " unit ",
3        " geometry ",
4        " level type ", " level 1",
5        " tau ", " stat type ", " stat val ",
6        " v_src ", " obs_type "))')

c*****
c  for each parameter and level, read obs from LLT DB
c*****
    dtg(1:10) = crdate_val(1:10)

    do i=1, arr_size
        write (0,('parm = ',a32)) verif(i)%param
        write (0,('unit = ',a10)) verif(i)%unit
        write (0,('type lvl = ',a10)) verif(i)%typlvl
        write (0,('lvl 1 = ',f8.1)) verif(i)%lvl_1
        write (0,('obs type = ',a15)) verif(i)%obs_type

        CALL LLTREAD(verif(i)%obs_type, verif(i)%obs_param,
2        verif(i)%lvl_1, verif(i)%odsetnm,
3        dtg, minlat,
4        maxlat, minlon,
5        maxlon, verif(i)%typlvl,
6        oblat, oblon,
7        nob, obval,
8        istat)

        write (0,('nob = ',i6)) nob

```

```

        if (istat /= 0 .or. nobs == 0) then
            go to 100
        end if

c*****
c    for each tau read the fcst, interpolate the fcst to obs
c    compute the stats and output the stats.
c*****
        do ltau = itau1, itaue, itauinc

            !determine the correct dtg table to read
            CALL DTGMOD(dtg, -ltau, cdtg, istat)
            if (istat /= 0) then
                go to 200
            end if

c*****
c    ISIS has tables for 00 and 12 only, therefore if we need
c    to read other tau model forecasts, e.g., 3, 6, 9, 15, 18, 21
c    etc., we need to read 12 hour old table
c    if cdtg ends with anything other than 00 or 12 then
c    use -12 ISIS table dtg
c*****
            if (cdtg(9:10) /= '00' .and. cdtg(9:10) /= '12') then
                CALL DTGMOD(tdtg, -12, cdtg, istat)
                if (istat /= 0) then
                    write *, 'DTGMOD error for tau ', ltau
                    go to 200
                end if
            else
                tdtg = cdtg
            end if
            write (0, ('"ISIS table dtg =",a10')) cdtg
            write (0, ('"tau =",i5')) ltau
            ftau = ltau

c*****
c    if everything is OK, then read the forecast
c    (gridded data) by calling ISIS GRD
c    ISIS grid does not have wnd_spd, therefore have to
c    read wnd_ucmp, wnd_vcmp then compute wnd_spd
c*****
            if (verif(i)%param == 'wnd_spd') then

c=====
c    read forecast for wnd_ucmp: ISIS grid data
c=====

                CALL GRD(modelname, geomname,
2                 verif(i)%dsetnm, 'wnd_ucmp',
3                 verif(i)%typ1vl, verif(i)%lvl_1,
4                 level_2, cdtg,
5                 ftau, verif(i)%unit,

```

```

6      fcstu,      title,
7      seclvl,     id,
8      iseq,       status)
      if (status /= 0) then
        go to 200
      end if

c=====
c      read forecast for wnd_vcmp: ISIS grid data
c=====

      CALL GRD(modelname, geomname,
2      verif(i)%dsetnm, 'wnd_vcmp',
3      verif(i)%typ1vl, verif(i)%1vl_1,
4      level_2,      cdtg,
5      ftau,         verif(i)%unit,
6      fcstv,        title,
7      seclvl,       id,
8      iseq,         status)
      if (status /= 0) then
        go to 200
      end if

c=====
c      first interpolate into another geometry if the second
c      geometry is requested
c=====

      if (geomname2 /= cnul .and. geomname2 /= geomname) then
        CALL INTGEOM(geomname, geomname2, ijmax, fcstu,
2      fcstu2, istat)
        if (istat == 0) fcstu = fcstu2
        CALL INTGEOM(geomname, geomname2, ijmax, fcstv,
2      fcstv2, istat)
        if (istat == 0) fcstv = fcstv2
        CALL GGRD(geomname2, ngeom, istat)
      end if

c=====
c      interpolate forecast into obs points, then
c      convert wnd_ucmp and wnd_vcmp to wnd_spd by calling
c      uv2df
c=====

      CALL F2OB(ngeom, fcstu, oblat, oblon, nob, fu, istat)
      CALL F2OB(ngeom, fcstv, oblat, oblon, nob, fv, istat)

c*****
c      if fcst had any missing value, f2ob fills fob with
c      -999 which needs to be disregarded in the conversion
c*****
      k = 0
      do j=1, nob
        if (fu(j) /= -999 .and. fv(j) /= 999) then
          k = k + 1

```

```

        newfu(k) = fu(j)
        newfv(k) = fv(j)
    end if
end do
CALL UV2DF(newfu, newfv, nobs, fdir, fob)

c*****
c    If the parameter is other than winds,
c    call GRD only once
c*****
    else

=====
c    read forecast: ISIS grid data
=====
    CALL GRD(modelname, geomname,
2        verif(i)%dsetnm, verif(i)%param,
3        verif(i)%typlvl, verif(i)%lvl_1,
4        level_2, cdtg,
5        ftau, verif(i)%unit,
6        fcst, title,
7        seclvl, id,
8        iseq, status)

    if (status /= 0) then
        go to 200
    end if

=====
c    first interpolate into another geometry if the second
c    geometry is requested
=====
    if (geomname2 /= cnul .and. geomname2 /= geomname) then
        CALL INTGEOM(geomname, geomname2, ijmax, fcst,
2        fcst2, istat)
        if (istat == 0) fcst = fcst2
        CALL GGRD(geomname2, ngeom, istat)
    end if

c*****
c    !interpolate fcst into obs pts
c*****
    CALL F2OB(ngeom, fcst, oblat, oblon, nobs, fob, istat)
    if (istat /= 0) then
        go to 200
    end if

    end if !param test

c*****
c    if fcst had any missing value, f2ob fills fob with
c    -999 which needs to be disregarded in the stat computations

```

```

c*****
      k = 0
      do j=1, nobs
        if (fob(j) /= -999) then
          k = k + 1
          newfob(k) = fob(j)
          newlat(k) = oblat(j)
          newlon(k) = oblon(j)
          newobs(k) = obval(j)
        end if
      end do
      write (0,('writing the 1st 15 lat,lon,obs,fcst.))')
      write (0,'(i5,4f10.2)') (n,newlat(n),newlon(n),newobs(n)
2      ,newfob(n), n=1,15)

c*****
      !compute the stats
      !for obs, disregard the projection
c*****

      nul_geom = 'NONE'

c*****
c      if there was no NAMLIST_FILE1, then use the stats
c      from NAMLIST_FILE2. For ocean models
c*****
      if (nstat == 0) then
        nstat = sfc_nstat
        do j = 1, nstat
          stats(j) = sfc_stats(j)
        end do
      end if

      if (geomname2 /= cnul) then
        outgeomname = geomname2
      else
        outgeomname = geomname
      end if

      do j = 1, nstat
        if (stats(j) == 'bias' .and. k /= 0) then
          CALL COMPUTE_BIAS(newfob,newobs,k,nul_geom,bias)
          !write the stats to the output file
          write (lstat, 1000) dtg, k, verif(i)%param,
2             verif(i)%unit,
3             outgeomname, verif(i)%typ1vl,
4             verif(i)%lvl_1, ltau,
5             stats(j), bias,
6             verif(i)%obs_type
          write (0,'(a30, 2a15,f10.2)') verif(i)%obs_type,
2             verif(i)%param, stats(j), bias

```

```

        else if (stats(j) == 'std' .and. k /= 0) then
            CALL COMPUTE_STD(newfob, newobs, k, nul_geom, std)
            write (lstat, 1000) dtg, k, verif(i)%param,
2                verif(i)%unit,
3                outgeomname, verif(i)%typlvl,
4                verif(i)%lvl_1, ltau,
5                stats(j), std,
6                verif(i)%obs_type
            write (0, '(a30, 2a15, f10.2)') verif(i)%obs_type,
2                verif(i)%param, stats(j), std

        else if (stats(j) == 'rms' .and. k /= 0) then
            CALL COMPUTE_RMS(newfob, newobs, k, nul_geom, rms)
            write (lstat, 1000) dtg, k, verif(i)%param,
2                verif(i)%unit,
3                outgeomname, verif(i)%typlvl,
4                verif(i)%lvl_1, ltau,
5                stats(j), rms,
6                verif(i)%obs_type
            write (0, '(a30, 2a15, f10.2)') verif(i)%obs_type,
2                verif(i)%param, stats(j), rms

c        else if (stats(j) == 'ancor' .and. k /= 0) then
c            CALL COMPUTE_RMS(newfob, newobs, k, nul_geom, ancor)
c            write (lstat, 1000) dtg, k, verif(i)%param,
c 2                verif(i)%unit,
c 3                outgeomname, verif(i)%typlvl,
c 4                verif(i)%lvl_1, ltau,
c 5                stats(j), ancor,
c 6                verif(i)%obs_type
c            write (0, '(a30, 2a15, f10.2)') verif(i)%obs_type,
c 2                verif(i)%param, stats(j), ancor

        !other stat types ...
        end if
    end do !do j=1,nstat
200    continue
    end do !ltau loop
100    continue
    end do ! i=1,arr_size loop

1000 format (a10,i6,1x,2a15,a30,a15,f8.2,i5,1x,a12,f8.2,' obs',1x,a15)
    close(lstat)
    CALL DBSTOP
    stop 'Normal End'
    CALL EXIT(0)
end

```

5. boundary.f90

```

subroutine boundary(ngeom, nrows, ncols, minlat,

```

```

2          maxlat, minlon, maxlon, istat)
C
C.....START PROLOGUE.....
C
C SCCS IDENTIFICATION: @(#)boundary.f90 1.1 04/24/98 /h/cm/library/mverif/src/sub/boundary.f90_v
C
C CONFIGURATION IDENTIFICATION: NONE
C
C MODULE NAME: boundary
C
C DESCRIPTION: This subroutine computes the minimum and maximum
C               latitude and longitude for reading observations
C               from ISIS LLT database.
C
C COPYRIGHT:      (c) 1996 FLENUMMETOCCEN
C                 U.S. GOVERNMENT DOMAIN
C                 ALL RIGHTS RESERVED
C
C CONTRACT NUMBER AND TITLE: N/A
C
C REFERENCES: NONE
C
C CLASSIFICATION: Unclassified
C
C RESTRICTIONS: NONE
C
C COMPUTER/OPERATING SYSTEM
C               DEPENDENCIES: Cray UNICOS
C
C LIBRARIES OF RESIDENCE: /a/ops/bin
C
C USAGE:
C   call boundary(igeom, nrows, ncols, minlat,
C                 maxlat, minlon, maxlon, istat)
C
C PARAMETERS:
C   Name      Type      Usage      Description
C   -----
C   NGEOM      INTEGER    INPUT    Geometry info.
C   NROWS      INTEGER    INPUT    No. of rows
C   NCOLS      INTEGER    INPUT    No. of columns
C   MINLAT     REAL       OUTPUT    Minimum latitude
C   MAXLAT     REAL       OUTPUT    Maximum latitude
C   MINLON     REAL       OUTPUT    Minimum longitude
C   MAXLON     REAL       OUTPUT    Maximum longitude
C   ISTAT      INTEGER    OUTPUT    Return status
C
C COMMON BLOCKS: N/A
C
C FILES: N/A
C
C DATA BASES: N/A

```

```

C
C NON-FILE INPUT/OUTPUT: N/A
C
C ERROR CONDITIONS:
C   CONDITION          ACTION
C   -----
C Error return from VXYLL  Print err message
C
C ADDITIONAL COMMENTS: NONE
C
C.....MAINTENANCE SECTION.....
C
C MODULES CALLED:
C   Name      Description
C   -----
C   MAXVAL    Returns maximum value from an array
C   MINVAL    Returns minimum value from an array
C   VXYLL     Computes arrays of lat/lon from arrays of x/y
C
C LOCAL VARIABLES AND      Structures are documented in detail
C   STRUCTURES:      where they are defined in the code
C                   within include files.
C
C METHOD: 1. Fill in the working arrays, x & y, with values for left,
C         right, bottom & top boundaries.
C         2. Call vxyl to get lat/lon from x & y.
C         3. Find the minimum & maximum latitude & longitude.
C
C INCLUDE FILES: NONE
C
C COMPILER DEPENDENCIES: f90
C
C COMPILE OPTIONS: -f fixed -c
C
C MAKEFILE: Located at /a/ops/app/mverif/src/sub/makeverobslib
C   UNICOS make
C
C RECORD OF CHANGES:
C
C <<CHANGE NOTICE>> Version 1.1 (29 Apr 1998) -- Kyongsuk Pace
C   Initial submission
C   Original Programmer: M.A. Rennick
C
C.....END PROLOGUE.....
C
C   implicit none

```

```

!*****
! parameters
!*****
integer, intent(in) :: ngeom ! geometry
integer, intent(in) :: nrow ! no of rows

```

```

integer, intent(in) :: ncols ! no of columns
real, intent(out) :: minlat ! min lat
real, intent(out) :: maxlat ! max lat
real, intent(out) :: minlon ! min lon
real, intent(out) :: maxlon ! max lon
integer, intent(out) :: istat ! return status

|*****
! local variables
|*****
integer :: len          ! array size
integer :: i            ! dummy loop var
integer :: n            ! temp var to hold array value
real :: x(2*nrows + 2*ncols) ! working array 1st dimension
real :: y(2*nrows + 2*ncols) ! working array 2nd dimension
real :: lat(2*nrows + 2*ncols) ! latitude array
real :: lon(2*nrows + 2*ncols) ! longitude array

|*****
! for left boundary
|*****
do i=1, nrows
  x(i) = 1 ! all x value at the left boundary is 1
  y(i) = i ! y value at the left boundary
end do

|*****
! for right boundary
|*****
n = 0 ! initialize the temp var
do i=nrows+1, 2*nrows
  x(i) = ncols ! all x value at the right boundary
  n = n + 1
  y(i) = n ! y value at the right boundary
end do

|*****
! for bottom boundary
|*****
n = 0 ! initialize the temp var
do i=2*nrows+1, 2*nrows+ncols
  n = n + 1
  x(i) = n ! all x value at the bottom boundary
  y(i) = 1 ! y value at the bottom boundary
end do

|*****
! for top boundary
|*****
n = 0 ! initialize the temp var
do i=2*nrows+ncols+1, 2*nrows+2*ncols
  n = n + 1

```

```

      x(i) = n    ! all x value at the bottom boundary
      y(i) = nrows ! y value at the bottom boundary
end do

```

```

!*****
!  call vxyl1 to get the lat/lon from x/y
!*****
      len = 2*nrows + 2*ncols
      call vxyl1(ngeom, len, x, y, 'd', lat, lon, istat)

      if (istat .eq. 0) then

!*****
!  get the min/max lat/lon
!*****
          minlat = minval(lat)
          maxlat = maxval(lat)
          minlon = minval(lon)
          maxlon = maxval(lon)

      else
          write (*,('vxyl1 returns istat =',i5)) istat
      end if

      return
end subroutine boundary

```

6. f2ob.f90

```

subroutine f2ob(igeom,field,obslat,obslon,nobs,fob,istat)
C
C.....START PROLOGUE.....
C
C SCCS IDENTIFICATION: @(#)f2ob.f90 1.1 04/24/98 /h/cm/library/mverif/src/sub/f2ob.f90_v
C
C CONFIGURATION IDENTIFICATION: NONE
C
C MODULE NAME: f2ob
C
C DESCRIPTION: This subroutine interpolates forecast field values to
C               observation locations.
C
C COPYRIGHT:      (c) 1998 FLENUMMETOCCEN
C                 U.S. GOVERNMENT DOMAIN
C                 ALL RIGHTS RESERVED
C
C CONTRACT NUMBER AND TITLE: N/A
C
C REFERENCES: NONE
C
C CLASSIFICATION: Unclassified

```

```

C
C RESTRICTIONS: NONE
C
C COMPUTER/OPERATING SYSTEM
C     DEPENDENCIES: Cray UNICOS
C
C LIBRARIES OF RESIDENCE: /a/ops/bin
C
C USAGE:
C   call f2ob(igeom,field,obslat,obslon,nobs,fob,istat)
C
C PARAMETERS:
C   Name      Type      Usage      Description
C   -----
C   IGEOM      INTEGER    INPUT    Geometry info.
C   FIELD      REAL       INPUT    Forecast array to interpolate
C   OBSLAT     REAL       INPUT    Obs. latitude
C   OBSLON     REAL       INPUT    Obs. longitude
C   NOBS       INTEGER    INPUT    No. of observations
C   FOB        REAL       OUTPUT   Fcst interpolated to obs array
C   ISTAT      INTEGER    OUTPUT   Return status
C
C COMMON BLOCKS: N/A
C
C FILES: N/A
C
C DATA BASES: N/A
C
C NON-FILE INPUT/OUTPUT: N/A
C
C ERROR CONDITIONS:
C   CONDITION      ACTION
C   -----
C   Unsuccessful getgeom    Print err message and exit
C
C ADDITIONAL COMMENTS: NONE
C
C.....MAINTENANCE SECTION.....
C
C MODULES CALLED:
C   Name      Description
C   -----
C   EXIT      System call that exits program
C   FINTRP    FORTRAN sub. that interpolates within a field to
C             obtain values at an array of points
C   GETGEOM   Gets geometry arguments to be used by other routines
C   IMAXCV    FORTRAN function to determine max. 1st dimension
C             of array
C   JMAXCV    FORTRAN function to determine max. 2nd dimension
C             of array
C   VLLXY     Computes arrays of x/y from arrays of lat/lon
C

```

```

C LOCAL VARIABLES AND      Structures are documented in detail
C   STRUCTURES:      where they are defined in the code
C                   within include files.
C
C METHOD: 1. Call getgeom.
C        2. If successful getgeom, determine max. 1st & 2nd dimensions
C           of the array using imaxcv/jmaxcv.
C        3. Call vlxy to convert from lat/lon to i,j.
C        4. Call fintrp to interpolate.
C
C INCLUDE FILES:
C   Name              Description
C   -----
C   V_DATA.H          contains the common variables for verobs
C
C COMPILER DEPENDENCIES: f90
C
C COMPILE OPTIONS: -f fixed -c
C
C MAKEFILE: Located at /a/ops/app/mverif/src/sub/makeverobslib
C             UNICOS make
C
C RECORD OF CHANGES:
C
C <<CHANGE NOTICE>> Version 1.1 (29 Apr 1998) -- Kyongsuk Pace
C   Initial submission
C   Original Programmer: MA Rennick
C
C.....END PROLOGUE.....
C

```

```

implicit none
include 'v_data.h'

```

```

!*****
!  Formal parameters
!*****
integer, intent(in) :: igeom      ! geom ptr from ggrd
integer, intent(in) :: nobs       ! number of obs to interpolate
real,  intent(in) :: field(ijmax) ! array to interpolate
real,  intent(in) :: obslat(maxobs) ! lat of report
real,  intent(in) :: obslon(maxobs) ! lon of report
real,  intent(out) :: fob(maxobs) ! interpolated array
integer, intent(out) :: istat      ! status

!*****
!  Local variables
!*****
integer :: imx, jmx
character(8) :: dsc
real :: dcol
real :: drow
real :: fx(maxobs)

```

```

real      :: fy(maxobs)
real      :: origi
real      :: origj
real      :: parm1, parm2, parm3

double precision :: olat
double precision :: olon

|*****
!  Functions
|*****
integer imaxcv, jmaxcv

CALL GETGEOM(igeom, prjnm, dsc, ncols, nrows, olat, olon,
2  origi, origj, dcol, drow, parm1, parm2, parm3, istat)

if (ncols*nrows .gt. ijmax) istat = 2

|*****
!  if successful getgeom, obtain max 1st and 2nd dimension of the array
!  and convert from lat/lon to i,j and interpolate
|*****
if (istat .eq. 0) then

    imx = imaxcv(ncols, nrows, dsc)
    jmx = jmaxcv(ncols, nrows, dsc)
    CALL VLLXY(igeom,nobs,obslat,obslon,'d',fx,fy,istat)
    if (istat .eq. 0) then
!      CALL FINTRP(fx,fy,nobs,field,imx,imx,jmx,0,0.,0.,0.,fob)
      CALL FINTRP(fx,fy,nobs,field,imx,imx,jmx,1,
2      bad_value, -999., -999., fob)
    else
      write (*,('vllxy returns istat =',i5)) istat
    end if

|*****
!  if unsuccessful getgeom, print the error msg
|*****
else

    write (*,('getgeom returns:'))
2    /" prjnam = ",a
3    /" dsc   = ",a
4    /" ncols = ",i5
5    /" nrows = ",i5
6    /" olat  = ",f8.2
7    /" olon  = ",f8.2
8    /" origi = ",f8.2
9    /" origj = ",f8.2
a    /" dcol  = ",f8.2
b    /" drow  = ",f8.2
c    /" parm1 = ",f8.2

```

```

d      /" parm2 = ",f8.2
e      /" parm3 = ",f8.2
f      /" istat = ",i5') prjnm,dsc,ncols,nrows,olat
g      ,olon,origi,origj,dcol,drow,parm1,parm2,parm3,istat

      if (istat .eq. 2) then
        write (*,('ERROR: Current geometry requires array "
2        "length ge",i10,"; ijmax =",i10)') ncols*nrows,ijmax
        CALL EXIT (1)
      end if

      end if ! (if istat == 0)

      return
      end subroutine f2ob

```

7. intgeom.f90

```

      subroutine intgeom(geomname1, geomname2, arr_size,
2      field1, field2, istat)
C
C.....START PROLOGUE.....
C
C SCCS IDENTIFICATION: @(#)intgeom.f90 1.1 04/24/98 /h/cm/library/mverif/src/sub/intgeom.f90_v
C
C CONFIGURATION IDENTIFICATION: NONE
C
C MODULE NAME: intgeom
C
C DESCRIPTION: Interpolates one geometry to another geometry for
C               spherical projection.
C
C COPYRIGHT:      (c) 1998 FLENUMMETOCEN
C                  U.S. GOVERNMENT DOMAIN
C                  ALL RIGHTS RESERVED
C
C CONTRACT NUMBER AND TITLE: N/A
C
C REFERENCES: NONE
C
C CLASSIFICATION: Unclassified
C
C RESTRICTIONS: NONE
C
C COMPUTER/OPERATING SYSTEM
C      DEPENDENCIES: Cray UNICOS
C
C LIBRARIES OF RESIDENCE: /a/ops/bin
C
C USAGE:
C  call intgeom(geomname1, geomname2,arr_size, field1, field2, istat)

```

```

C
C PARAMETERS:
C   Name      Type      Usage      Description
C   -----
C   GEOMNAME1  CHAR*32   INPUT   GEOMETRY TO INTERPOLATE FROM
C   GEOMNAME2  CHAR*32   INPUT   GEOMETRY TO INTERPOLATE TO
C   ARR_SIZE   INTEGER   INPUT   ARRAY_SIZE
C   FIELD1     REAL(ijmax) INPUT   ARRAY TO INTERPOLATE FROM
C   FIELD2     REAL(ijmax) OUTPUT  INTERPOLATED ARRAY
C   ISTAT      INTEGER   OUTPUT  STATUS
C
C COMMON BLOCKS: N/A
C
C FILES: NONE
C
C DATA BASES: $META_GRID_DB
C   Name      Table      Usage      Description
C   -----
C
C NON-FILE INPUT/OUTPUT: NONE
C
C ERROR CONDITIONS:
C   CONDITION      ACTION
C   -----
C Error return from GGRD   Print err message & exit
C Error return from GETGEOM Print err message & exit
C Error return from VXYLL  Print err message & exit
C Error return from CHGEOM Print err message & exit
C
C ADDITIONAL COMMENTS: NONE
C
C.....MAINTENANCE SECTION.....
C
C MODULES CALLED:
C   Name      Description
C   -----
C   CHGEOM    CHANGES THE GEOMETRY
C   EXIT      System call that exits program
C   GETGEOM   Gets geometry arguments to be used by other routines
C   GGRD      Returns ISIS info. on given geometry
C   IMAXCV    Selects the first dimension of a field
C   JMAXCV    Selects the second dimension of a field
C   VXYLL     Converts i/j to lat/lon
C
C LOCAL VARIABLES AND
C   STRUCTURES:
C
C   Name      Type      Description
C   -----
C   CNCOLS    INTEGER   column count (getgeom, geomname1)
C   CNROWS    INTEGER   row count (getgeom, geomname1)
C   CORIGLAT  DOUBLE    latitude of origin (getgeom,

```

C
C CORIGLON DOUBLE longitude of origin (getgeom, geomname1)
C CORIGX REAL x coordinate of origin (getgeom, geomname1)
C CORIGY REAL y coordinate of origin (getgeom, geomname1)
C CPARM1 REAL geometry parameter #1 (getgeom, geomname1)
C CPARM2 REAL geometry parameter #2 (getgeom, geomname1)
C CPARM3 REAL geometry parameter #3 (getgeom, geomname1)
C CPRJNAM CHAR*24 projection name (getgeom, geomname1)
C CSTDESC CHAR*24 storage description (getgeom, geomname1)
C CXINTDIS REAL interval distance between columns (getgeom, geomname1)
C CYINTDIS REAL interval distance between rows (getgeom, geomname1)
C FILVAL REAL chgeom parameter
C FLAT REAL latitude array
C FLON REAL longitude array
C FVALI REAL chgeom parameter
C FVALO REAL chgeom parameter
C FW1 REAL work array for chgeom
C FW2 REAL work array for chgeom
C GEOMI INTEGER data structure containing the input
C grid definition (chgeom)
C GGEOM CHAR*32 data structure containing the output
C grid definition (chgeom)
C GMODEL CHAR*32 NOGAPS model
C I INTEGER counter
C IFLAGI INTEGER input field flag (chgeom)
C IMAX INTEGER first dimension of an array
C ISFLG INTEGER chgeom flag
C ISTGR INTEGER stagger flag (chgeom)
C IVEC INTEGER vector flag (chgeom)
C IWRP INTEGER wrap flag (chgeom)
C J INTEGER counter
C JMAX INTEGER second dimension of an array
C LAFLAG INTEGER land average flag (chgeom)
C LAPASS INTEGER number of passes (chgeom)
C LASRCH INTEGER number of points to search (chgeom)
C LAVAL REAL values in the field (chgeom)
C LEN INTEGER total number of element in an array
C LSTATS INTEGER output unit
C NCOLS INTEGER column count
C NGEOM INTEGER data structure containing the input
C grid definition (chgeom)
C NRFCST REAL NORAPS forecast field
C NRFCSTU REAL NORAPS wind_u forecast
C NRFCSTV REAL NORAPS wind_v forecast
C NROWS INTEGER row count
C ORIGLAT DOUBLE latitude of origin
C ORIGLON DOUBLE longitude of origin
C ORIGX REAL x coordinate of origin
C ORIGY REAL y coordinate of origin
C PARM1 REAL geometry parameter #1
C PARM2 REAL geometry parameter #2
C PARM3 REAL geometry parameter #3

```

C   PRJNAM   CHAR*24   projection name
C   STDESC   CHAR*24   Storage description
C   XINTDIS   REAL     Interval distance between columns
C   YINTDIS   REAL     Interval distance between rows
C
C METHOD:
c   1. Get ISIS information on geomname1 by calling GGRD and
c       GETGEOM.
c   2. Call GGRD and GETGEOM on the geomname2.
c   3. Test for 'spherical' projection.
c   4. Find the first and second dimension of the geomname2.
c   5. Convert i/j to lat/lon by calling vxyl.
c   6. Interpolate field1 to field2 by calling CHGEOM.
C
C INCLUDE FILES: NONE
C
C COMPILER DEPENDENCIES: f90
C
C COMPILE OPTIONS: -f fixed -c
C
C MAKEFILE: Located at /a/ops/app/mverif/src/sub/makeverobslib
C   UNICOS make
C
C RECORD OF CHANGES:
C
C <<CHANGE NOTICE>> Version 1.1 (29 Apr 1998) -- Kyongsuk
C   Initial submission
C
C.....END PROLOGUE.....
C
implicit none

```

```

c*****
c   Formal parameters
c*****
character(32), Intent(in) :: geomname1    ! first geometry name
character(32), Intent(in) :: geomname2    ! second geom name
integer,    Intent(in) :: arr_size    ! array size
real,      Intent(in) :: field1(arr_size)!array to interpolate
real              :: field2(arr_size)!interpolated array
integer              :: istat    ! status

c*****
c   Local variables
c*****
character*1 uv
character*24 stdesc, cstdesc
character*24 prjnam, cprjnam
character*32 gmodel, ggeom

integer i, j
integer lstats, ngeom

```

```

integer geomi, ncols, nrows
integer cncols, cnrows, im, jm, imax, jmax, len
integer ivec, iwrp, istgr, iflagi, laflag, lasrch, lapass, isflg

real  fvali, fvalo, filval, laval
real  origx, origy, parm1, parm2, parm3, xintdis, yintdis
real  corigx, corigy, cparm1, cparm2, cparm3, cxintdis, cyintdis

parameter (im = 360)
parameter (jm = 181)
real  fw1(im,jm), fw2(im,jm), flat(im,jm), flon(im,jm)

double precision origlat, origlon, coriglat, coriglon
parameter (uv = 'd') ! for vxyl 'd' means in degrees

```

```

c*****
c  Functions
c*****
integer imaxcv, jmaxcv

c*****
c  get ISIS info on the geometry1 by calling GGRD
c  and GETGEOM
c*****
call GGRD(geomname1, geomi, istat)
if (istat .ne. 0) then
  write (0,('ggrd for geomname1 returns istat =',i5)) istat
  call exit(11)
end if
call getgeom(geomi, cprjnam, cstdesc, cncols, cnrows, coriglat,
2 coriglon, corigx, corigy, cxintdis, cyintdis, cparm1,
3 cparm2, cparm3, istat)

if (istat .ne. 0) then
  write (0,('getgeom for geomname returns istat =',i5)) istat
  call exit(11)
end if

if (cprjnam .ne. 'spherical') then
  write(0,('projection name for geomname1 is not spherical'))
  call exit(11)
end if

c*****
c  get ISIS info on the geometry2 by calling GGRD
c*****
call ggrd(geomname2,ngeom,istat)
if (istat .ne. 0) then
  write (0,('ggrd for geomname2 returns istat =',i5)) istat
  call exit(11)
end if

```

```

    call getgeom(ngeom, prjnam, stdesc, ncols, nrows, origlat,
2      origlon, origx, origy, xintdis, yintdis, parm1, parm2,
3      parm3, istat)

    if (istat .ne. 0) then
      write(0,('getgeom for geomname2 returns istat =',i5))
2      istat
      call exit(11)
    end if

    if (prjnam .ne. 'spherical') then
      write(0,('projection name for geomname2 is not spherical'))
      call exit(11)
    end if

c=====
c  convert the i,j to lat/lon
c  ref: /a/library/omsp/chgeom/src/sub/chgeom.f
c  flat=array of latitudes from vxyl for new 2-D array geom0
c  flon=array of longitudes from vxyl for new 2-D array geom0
c=====
    imax=imaxcv(ncols,nrows,stdesc)
    jmax=jmaxcv(ncols,nrows,stdesc)
    do j=1, jmax
      do i=1, imax
        fw1(i,j) = float(i)
        fw2(i,j) = float(j)
      enddo
    enddo

    len = imax*jmax
    call vxyl(ngeom, len, fw1, fw2, uv, flat, flon, istat)
    if (istat .ne. 0) then
      write(0,('VXYLL error'))
      call exit(11)
    endif

c=====
c  set some of the chgeom parameter values (ref: chgeom write-up)
c=====
    ivec = 1
    iwrp = 1
    istgr = 0
    iflagi = 0
    fvali = 0.0
    fvalo = 0.0
    laflag = 0
    lasrch = 20
    laval = 0.0
    lapass = 0
    filval = 0.0
    isflg = 0

```

```

c=====
c  change the field1 to field2
c=====
      call chgeom(field1, field1, cncols, cnrows, geom1, imax, jmax,
1       ngeom, ivec, iwrp, istgr, iflagi, fvali, fvalo,
2       laflag, lasrch, laval, lapass, filval, isflg,
3       fw1, fw2, field2, field2, istat)
      if (istat .ne. 0) then
        write(0,('CHGEOM error in intgeom'))
        call exit(11)
      endif

      return
      end subroutine intgeom

```

8. uv2df.f90

```

      subroutine uv2df(u, v, n, dir, spd)
C
C.....START PROLOGUE.....
C
C SCCS IDENTIFICATION: @(#)uv2df.f90 1.1 04/24/98 /h/cm/library/mverif/src/sub/uv2df.f90_v
C
C CONFIGURATION IDENTIFICATION: NONE
C
C MODULE NAME: uv2df
C
C DESCRIPTION: This subroutine converts u/v components to a
C              field of direction/speed (dd/ff).
C
C COPYRIGHT:      (c) 1996 FLENUMMETOCCEN
C                U.S. GOVERNMENT DOMAIN
C                ALL RIGHTS RESERVED
C
C CONTRACT NUMBER AND TITLE: N/A
C
C REFERENCES: NONE
C
C CLASSIFICATION: Unclassified
C
C RESTRICTIONS: NONE
C
C COMPUTER/OPERATING SYSTEM
C              DEPENDENCIES: Cray UNICOS
C
C LIBRARIES OF RESIDENCE: /a/ops/bin
C
C USAGE:
C  call uv2df(u, v, n, dir, spd)
C
C PARAMETERS:

```

```

C   Name      Type      Usage      Description
C   -----
C   U          REAL      INPUT      Wind u-comp fct interpolated to obs
C   V          REAL      INPUT      Wind v-comp fct interpolated to obs
C   N          INTEGER    INPUT      No. of observations
C   DIR        REAL      OUTPUT     Converted direction array
C   SPD        REAL      OUTPUT     Converted speed array
C
C COMMON BLOCKS: N/A
C
C FILES: N/A
C
C DATA BASES: N/A
C
C NON-FILE INPUT/OUTPUT: N/A
C
C ERROR CONDITIONS: N/A
C
C ADDITIONAL COMMENTS: NONE
C
C.....MAINTENANCE SECTION.....
C
C MODULES CALLED: N/A
C
C LOCAL VARIABLES AND      Structures are documented in detail
C   STRUCTURES:      where they are defined in the code
C                   within include files.
C   Name      Type      Description
C   -----
C   I          INTEGER    Counter
C   R2D        REAL      45.0 / atan(1.0)
C
C METHOD:
C   Convert u/v to dir/spd by using simple trigonometric functions.
C
C INCLUDE FILES: NONE
C
C COMPILER DEPENDENCIES: f90
C
C COMPILE OPTIONS: -f fixed -c
C
C MAKEFILE: Located at /a/ops/app/mverif/src/sub/makeverobslib
C   UNICOS make
C
C RECORD OF CHANGES:
C
C <<CHANGE NOTICE>> Version 1.1 (29 Apr 1998) -- Kyongsuk Pace
C   Initial submission
C   Original Programmer: M.A. Rennick
C
C.....END PROLOGUE.....
C

```

implicit none

```
c*****
c  formal parameters
c*****
  real u(n), v(n), dir(n), spd(n)
  integer n

c*****
c  local variables
c*****
  integer i
  real r2d
  real badval
  parameter(badval=1.E+10)

  r2d = 45.0 / atan(1.0)

  do i = 1, n
    if (u(i) .eq. badval) then
      dir(i) = 999.0
      spd(i) = 999.0
    else
      if (u(i) .eq. 0.0) then
        u(i) = 1.0e-6
      end if
      dir(i) = 270.0 - r2d * atan2(v(i), u(i))
      if (dir(i) .gt. 360.0) then
        dir(i) = dir(i) - 360.0
      end if
      spd(i) = sqrt(u(i)*u(i) + v(i)*v(i))
    end if
  end do

  return
end subroutine uv2df
```

9. lltread.f90

```
subroutine lltread(seq_type, param, lvl, dsetnm,
2      dtg, minlat, maxlat, minlon,
3      maxlon, typlvl, obslat, obslon,
4      nobs, obsval, istat)
C
C.....START PROLOGUE.....
C
C SCCS IDENTIFICATION: @(#)lltread.f90 1.1 04/24/98 /h/cm/library/mverif/src/sub/lltread.f90_v
C
C CONFIGURATION IDENTIFICATION: NONE
C
C MODULE NAME: lltread
```

C
 C DESCRIPTION: This module calls the appropriate latitude-
 C longitude-time (llt) read module based upon
 C the user specified llt sequence type(s).
 C There are separate read modules for each sequence
 C type because of the different include files and
 C data structure in the ISIS for the
 C different observation types.
 C
 C COPYRIGHT: (c) 1996 FLENUMMETOCCEN
 C U.S. GOVERNMENT DOMAIN
 C ALL RIGHTS RESERVED
 C
 C CONTRACT NUMBER AND TITLE: N/A
 C
 C REFERENCES: ISIS LLT User's Manual
 C
 C CLASSIFICATION: Unclassified
 C
 C RESTRICTIONS: NONE
 C
 C COMPUTER/OPERATING SYSTEM
 C DEPENDENCIES: Cray UNICOS
 C
 C LIBRARIES OF RESIDENCE: /a/ops/bin
 C
 C USAGE:
 C call llthread(seq_type, param, lvl, dsetnm, dtg,
 C minlat, maxlat, minlon, maxlon, typlvl,
 C obslat, obslon, nob, obsval, istat)
 C
 C PARAMETERS:

Name	Type	Usage	Description
SEQ_TYPE	char*24	INPUT	LLT obs type
PARAM	CHAR*32	INPUT	Parameter to read eg. air_temp
LVL	REAL	INPUT	Pressure level
DSETNM	CHAR*24	INPUT	ISIS dataset name
DTG	CHAR*10	INPUT	Date Time Group for read
MINLAT	REAL	INPUT	Minimum latitude of the area
MAXLAT	REAL	INPUT	Maximum latitude of the area
MINLON	REAL	INPUT	Minimum longitude of the area
MAXLON	REAL	INPUT	Maximum longitude of the area
TYPLVL	CHAR*24	INPUT	level type
OBSLAT	REAL(maxobs)	OUTPUT	Observation latitude
OBSLON	REAL(maxobs)	OUTPUT	Observation longitude
NOBS	INTEGER	OUTPUT	No of good obs read
OBSVAL	REAL(maxobs)	OUTPUT	Observed parameter value
ISTAT	INTEGER	OUTPUT	Status

 C
 C COMMON BLOCKS: N/A
 C

```

C FILES: NONE
C
C DATA BASES: NONE
C
C NON-FILE INPUT/OUTPUT: N/A
C
C ERROR CONDITIONS:
C   CONDITION          ACTION
C   -----
C   Numobs exceeds maxobs   Print err message & return
C   (istat = -1)
C
C ADDITIONAL COMMENTS: NONE
C
C.....MAINTENANCE SECTION.....
C
C MODULES CALLED:
C   Name          Description
C   -----
C   RAOB_QC_READ    reads ISIS raob_qc llt data
C   SFC_LAND_READ    reads ISIS surface land llt data
C   SFC_SHIP_READ    reads ISIS surface ship met llt data
C   SFC_SHIP_MET_QC_READ reads ISIS sfc ship met qc llt data
C   ALTY_READ        reads ISIS alty llt data
C
C LOCAL VARIABLES AND      Structures are documented in detail
C   STRUCTURES:      where they are defined in the code
C                   within include files.
C
C INCLUDE FILES:
C   Name          Description
C   -----
C   v_data.h      common variables used for verobs
C
C COMPILER DEPENDENCIES: f90
C
C COMPILE OPTIONS: -f fixed -c
C
C MAKEFILE: Located at /a/ops/app/mverif/src/sub/makeverobslib
C   UNICOS make
C
C RECORD OF CHANGES:
C
C <<CHANGE NOTICE>> Version 1.1 (29 Apr 1998) -- Kyongsuk Pace
C   Initial submission
C
C.....END PROLOGUE.....
C
C   implicit none
C   include 'v_data.h'
C*****

```

```

c  Formal parameters
c*****
character(24), intent(in) :: seq_type
Character(32), Intent(in) :: PARAM ! parameter
Real, Intent(in) :: lvl ! level
CHARACTER(24), INTENT(IN) :: DSETNM ! Data set name used.
CHARACTER(10), INTENT(IN) :: DTG ! Date Time Group for read.
REAL, INTENT(IN) :: MINLAT ! South latitude boundary.
REAL, INTENT(IN) :: MAXLAT ! North latitude boundary.
REAL, INTENT(IN) :: MINLON ! West longitude boundary.
REAL, INTENT(IN) :: MAXLON ! East longitude boundary.
CHARACTER(24), INTENT(IN) :: TYPLVL ! type level, e.g, isbr_lvl
Real :: obslat(maxobs) ! obs lat
Real :: obslon(maxobs) ! obs lon
Real :: obsval(maxobs) ! obs data value
integer :: nobobs ! numober of obs
integer :: istat

c*****
! Local variables used as arguments for LLT read subroutines:
c*****
CHARACTER(8) :: VRSNNAM ! Version of ISIS software used.
CHARACTER(8) :: SECLVL ! 7 character security classification level.
REAL :: HR ! Hour cited in the report.
CHARACTER(16) :: MINDTG ! Minmum date and time group to read.
CHARACTER(16) :: MAXDTG ! Maximum date and time group to read.
CHARACTER(16) :: NEW_DTG ! -12 DTG if current not found
REAL :: MINHR ! Minmum hour to read.
REAL :: MAXHR ! Maximum hour to read.

c*****
c  Local variables
c*****
integer :: i

REAL :: PLAT ! Point latitude
REAL :: PLON ! Point longitude
REAL :: DISTANCE ! Radius of circle centered
! at PLAT/PLON
CHARACTER(5) :: BLOCK_STATION(10)
CHARACTER(20) :: STATION_NAME(10) ! International blksta #
Character(20) :: stn_name

vrsnnam = '*'
seclvl = 'UNCLASS'

c*****
c  initialize the arrays oblat, oblon, obval
c*****
do i=1,maxobs
  obslat(i) = bad_value
  obslon(i) = bad_value

```

```

        obsval(i) = bad_value
    end do

```

```

c*****
c  RAOB_QC LLT (nogaps, noraps, coamps)
c*****
    if (seq_type == 'raob_qc') then
        CALL RAOB_QC_READ(vrsnnam, dsetnm, seclvl,
2           dtg, param, lvl,
3           minlat, maxlat, minlon,
4           maxlon, obslat, obslon,
5           nobs, obsval, istat)

c*****
c  sfc_lnd (nogaps, noraps, coamps)
c*****
    else if (seq_type == 'sfc_lnd') then
        CALL SFC_LND_READ(vrsnnam, dsetnm, seclvl,
2           dtg, param, lvl,
3           minlat, maxlat, minlon,
4           maxlon, obslat, obslon,
5           nobs, obsval, istat)

c*****
c  sfc_ship (wam_global, otis_global)
c*****
    else if (seq_type == 'sfc_ship') then
        CALL SFC_SHIP_READ(vrsnnam, dsetnm, seclvl,
2           dtg, param, lvl,
3           minlat, maxlat, minlon,
4           maxlon, obslat, obslon,
5           nobs, obsval, istat)

c*****
c  sfc_ship_met_qc (nogaps, noraps, coamps)
c*****
    else if (seq_type == 'sfc_ship_met_qc') then
        CALL SFC_SHIP_MET_QC_READ
2           (vrsnnam, dsetnm, seclvl,
3           dtg, param, lvl,
4           minlat, maxlat, minlon,
5           maxlon, obslat, obslon,
6           nobs, obsval, istat)

c*****
c  alty (wam_global)
c*****
    else if (seq_type == 'alty') then
        CALL ALT_Y_READ( vrsnnam, dsetnm, seclvl,
2           dtg, param, lvl,
3           minlat, maxlat, minlon,
4           maxlon, obslat, obslon,

```

```

5          nob,      obsval,      istat)

c  other seq_types

end if

return
end subroutine lltread

```

10. alty.f90

```

subroutine alty_read(vrsnam, dsetnm, seclvl, dtg, param,
2          lvl, minlat, maxlat, minlon, maxlon,
3          lat, lon, nob, obs, istat)
C
C.....START PROLOGUE.....
C
C SCCS IDENTIFICATION: @(#)alty.f90 1.1 04/24/98 /h/cm/library/mverif/src/sub/alty.f90_v
C
C CONFIGURATION IDENTIFICATION: NONE
C
C MODULE NAME: alty_read
C
C DESCRIPTION: subroutine to read the alty data and pick
C              out the obs data for the given parameter
C
C COPYRIGHT:      (c) 1998 FLENUMMETOCEN
C                U.S. GOVERNMENT DOMAIN
C                ALL RIGHTS RESERVED
C
C CONTRACT NUMBER AND TITLE: N/A
C
C REFERENCES: NONE
C
C CLASSIFICATION: Unclassified
C
C RESTRICTIONS: NONE
C
C COMPUTER/OPERATING SYSTEM
C              DEPENDENCIES: Cray UNICOS
C
C LIBRARIES OF RESIDENCE: /a/ops/bin
C
C USAGE:
C  call alty_read( vrsnam, dsetnm, seclvl, dtg, param,
C                  lvl, minlat, maxlat, minlon, maxlon,
C                  lat, lon, nob, obs, istat)
C
C PARAMETERS:
C  Name      Type      Usage      Description
C  -----

```

```

C VRSNNAM   CHAR*8   INPUT  llt version name
C DSETNM    CHAR*24   INPUT  data set name
C SECLVL    CHAR*8    INPUT  classification
C DTG       CHAR*10   INPUT  date time group for read
C PARAM     CHAR*32   INPUT  parameter
C LVL       REAL      INPUT  level type
C MINLAT    REAL      INPUT  minimum latitude
C MAXLAT    REAL      INPUT  maximum latitude
C MINLON    REAL      INPUT  minimum longitude
C MAXLON    REAL      INPUT  maximum longitude
C LAT       REAL(size) OUTPUT obs latitude
C LON       REAL(size) OUTPUT obs longitude
C NOBS      INTEGER   OUTPUT  number of obs
C OBS       REAL(size) OUTPUT  obs value
C ISTAT     INTEGER   OUTPUT  return status
C
C COMMON BLOCKS: N/A
C
C FILES: None
C
C DATA BASES: ISIS LLT_DB
C   Name      Table  Usage      Description
C   -----
C   alty      ALTY    IN        alty obs
C
C NON-FILE INPUT/OUTPUT: N/A
C
C ERROR CONDITIONS:
C   CONDITION      ACTION
C   -----
C   DTG error      Print err message & return
C   Error return from LRD  Print err message
C   Error return from LCLOS  Print err message
C
C ADDITIONAL COMMENTS: NONE
C
C.....MAINTENANCE SECTION.....
C
C MODULES CALLED:
C   Name      Description
C   -----
C   LCLOS     ISIS LLT close
C   LEN_TRIM  Determines the length of a string
C   LRD       ISIS LLT read
C   TRIM      Removes the trailing blanks
C
C LOCAL VARIABLES AND Structures are documented in detail
C   STRUCTURES:  where they are defined in the code
C               within include files.
C
C METHOD:
C   Set seq_type to 'alty'

```

```

C   See raob_qc_read for the rest.
C
C INCLUDE FILES:
C   Name                Description
C   -----
C   ALTY.H              alty header file
C
C COMPILER DEPENDENCIES: f90
C
C COMPILE OPTIONS: -f fixed -c
C
C MAKEFILE: Located at /a/ops/app/mverif/src/sub/makeverobslib
C   UNICOS make
C
C RECORD OF CHANGES:
C
C <<CHANGE NOTICE>> Version 1.1 (29 Apr 1998) -- Kyongsuk Pace
C   Initial submission
C
C .....END PROLOGUE.....
C
C   implicit none
C   include 'v_data.h'
C   include 'ALTY.H'

```

```

c*****
c   formal parameters
c*****
c   character(8), intent(in) :: vrsnnam
c   character(24), intent(in) :: dsetnm
c   character(8), intent(in) :: seclvl
c   character(10), intent(in) :: dtg
c   character(32), intent(in) :: param
c   real, intent(in) :: lvl
c   real, intent(in) :: minlat
c   real, intent(in) :: maxlat
c   real, intent(in) :: minlon
c   real, intent(in) :: maxlon
c   real :: lat(size)
c   real :: lon(size)
c   integer :: nob
c   real :: obs(size)
c   integer :: istat

c*****
c   local variables used as arguments for LRD:
c*****
c   character(24) :: seq_type
c   real :: hr
c   CHARACTER(16) :: MINDTG ! Minmum date and time group to read.
c   CHARACTER(16) :: MAXDTG ! Maximum date and time group to read.
c   REAL :: MINHR ! Minmum hour to read.

```

```

REAL      :: MAXHR      ! Maximum hour to read.
CHARACTER(16) :: RSN_IN  ! Reporting source name.
REAL      :: FCST_IN    ! Desired forecast "TAU".
CHARACTER(24) :: MINUPTM ! Minimum update time.
CHARACTER(56) :: REMARKS ! Description of data/assoc. record.
CHARACTER(16) :: RPT_DTG ! Actual date & time group for report.
REAL      :: RPT_HR     ! Reported hour read.
REAL      :: RPT_LAT    ! Reported latitude read.
REAL      :: RPT_LON    ! Reported longitude read.
CHARACTER(16) :: RPT_RSN ! Reported longitude read.
REAL      :: RPT_FCST    ! Reported "TAU" or forecast time.
CHARACTER(24) :: RPT_CRET ! Record creation date.
CHARACTER(24) :: RPT_UDT ! Report's last update time.
INTEGER     :: BUFLAG    ! 0 => Input is in FBUFF
                ! 1 => Input is in Ibuff
INTEGER     :: LLT_ID    ! Unique database LLT identifier for each
                ! dataset.
INTEGER     :: BLKSEQID  ! Unique database LLT block identifier.
INTEGER     :: RECSEQID  ! Unique database LLT record identifier.

TYPE(alty_int):: Ibuff    ! integer record structure.
TYPE(alty):: Fbuff       ! Real record structure.

c*****
c  Arguments for LCLOS (that get "*" for values).
c*****
CHARACTER(24) :: SEQTYPE_X ! Report type.
CHARACTER(8)  :: VRSNNAM_X ! Version of ISIS software used.
CHARACTER(24) :: DSETNAM_X ! Data set name used.
CHARACTER(8)  :: SECLVL_X  ! 7 character security
                ! classification level.
CHARACTER(16) :: DTG_X     ! Date Time Group for write.

c*****
c  Other local variables
c*****
integer :: levels
integer :: status, i, status2

seq_type = 'alty' ! Report type
istat = 0

!*****
!  Set up date and time group in YYYYMMDDHH format in DTG.
!*****
IF ( LEN_TRIM(DTG) == 10 ) THEN
  READ (UNIT=DTG(9:10),FMT=(F2.0),IOSTAT=STATUS) HR
  IF ( STATUS == 0 ) THEN
    IF ( HR < 12. ) THEN
      HR = 0.
    ELSE
      HR = 12.
    
```

```

        END IF
    ELSE
        WRITE *, ' Cannot read hour "', DTG(9:10),
2         ' from date & time group ', TRIM(DTG)
        istat = -1
        RETURN
    END IF
ELSE
    STATUS = 10
    WRITE *, 'alty: Got date and time group ',
2     TRIM(DTG), ' of length ',
3     LEN_TRIM(DTG), ' but expected length == 10.'
    istat = -1
    RETURN
END IF

!*****
!  Set the input parameters used to get a read-back value.
!*****
MINDTG = DTG;   MAXDTG = DTG
MINHR  = HR;    MAXHR  = HR+11.999
RSN_IN = '*'
FCST_IN = 0.0  ! Report forecast period or Tau (normal = 0.0)
MINUPTM = '*'
BUFFLAG = 0    ! Want (both) floating (and integer).

!*****
!  get the data from LLT db
!*****
I = 0
nobs = 0

DO WHILE ( STATUS == 0 )
    CALL LRD(seq_type, vrsnam, dsetnm, SECLVL,
2     MINDTG, MINHR, MAXDTG, MAXHR,
3     MINLAT, MAXLAT, MINLON, MAXLON,
4     RSN_IN, FCST_IN, MINUPTM, BUFFLAG,
5     RPT_DTG, RPT_HR, RPT_LAT, RPT_LON,
6     RPT_RSN, RPT_FCST, RPT_CRETM, RPT_UDT,
7     LLT_ID, BLKSEQID, RECSEQID,
8     Ibuff, Fbuff, STATUS )
    IF ( STATUS /= 0 ) THEN
        IF ( STATUS /= 100 ) THEN
            ! Ignore normal no-more-data return code
            WRITE *, ' Read from ISIS failed. Code = ', STATUS, ' '
            istat = status
        END IF
    ELSE ! successful LRD
        I = I + 1
    END IF
END DO

c*****
c    pick out the relevant info and fill the array

```

```

c*****
      if (param == 'sig_wav_ht') then
        if (fbuff % sig_wav_ht < check_val) then
          nobs = nobs + 1
          lat(nobs) = fbuff % crse_lat
          lon(nobs) = fbuff % crse_lon
          obs(nobs) = fbuff % sig_wav_ht
        end if

      else if (param == 'wnd_spd') then
        if (fbuff % wnd_spd < check_val) then
          nobs = nobs + 1
          lat(nobs) = fbuff % crse_lat
          lon(nobs) = fbuff % crse_lon
          obs(nobs) = fbuff % wnd_spd
        end if

      end if ! param
    end if ! OK status
  END DO ! i loop

  IF ( STATUS == 100 ) STATUS = 0
  write *, ''
  WRITE *, ' Called LRD ', I, ' times.'
  write *, ' Read ', nobs, ' obs of sequence type ', TRIM(SEQ_TYPE),
2      ''

!*****
! Close the dataset (now open for reading) again.
!*****
  SEQTYPE_X = '*'; VRSNNAM_X = '*'; DSETNAM_X = '*'
  SECLVL_X = '*'; DTG_X = '*'

  CALL LCLOS(SEQTYPE_X, VRSNNAM_X, DSETNAM_X,
2      SECLVL_X, DTG_X, STATUS2)
  IF ( STATUS2 /= 0 ) THEN
    WRITE *, ' Could not close ISIS table. ',
2      ' Error code is ', STATUS2, ''
  END IF

  return
end subroutine alty_read

```

11. raob_qc.f90

```

      subroutine raob_qc_read(vrsnnam, dsetnm, seclvl, dtg, param,
2          lvl, minlat, maxlat, minlon, maxlon,
3          lat, lon, kfinal, obs, istat)
c
c.....START PROLOGUE.....
c

```

```

c SCCS IDENTIFICATION: @(#)raob_qc.f90 1.1 04/24/98 /h/cm/library/mverif/src/sub/raob_qc.f90_v
c
c CONFIGURATION IDENTIFICATION: NONE
c
c MODULE NAME: raob_qc_read
c
c DESCRIPTION: subroutine to read the raob_qc data and pick
c               out the obs data for the given parameter
c
c COPYRIGHT:      (c) 1996 FLENUMMETOCCEN
c                  U.S. GOVERNMENT DOMAIN
c                  ALL RIGHTS RESERVED
c
c CONTRACT NUMBER AND TITLE: N/A
c
c REFERENCES: LLT User's Manual
c
c CLASSIFICATION: Unclassified
c
c RESTRICTIONS: NONE
c
c COMPUTER/OPERATING SYSTEM
c               DEPENDENCIES: Cray UNICOS
c
c LIBRARIES OF RESIDENCE: /a/ops/bin
c
c USAGE:
c   call raob_qc_read(vrsnnam, dsetnm, seclvl, dtg, param,
c                     level, minlat, maxlat, minlon, maxlon,
c                     lat, lon, kfinal, obs, istat)
c
c PARAMETERS:
c   Name      Type      Usage      Description
c   -----
c   VRSNNAM    INTEGER   INPUT    Version name
c   DSETNM     CHAR*24   INPUT    ISIS dataset name
c   SECLVL     CHAR*8    INPUT    Security level
c   DTG        CHAR*10   INPUT    Date Time Group for read
c   PARAM      CHAR*32   INPUT    Parameter to read eg. air_temp
c   LVL        REAL      INPUT    Pressure level
c   MINLAT     REAL      INPUT    Minimum latitude of the area
c   MAXLAT     REAL      INPUT    Maximum latitude of the area
c   MINLON     REAL      INPUT    Minimum longitude of the area
c   MAXLON     REAL      INPUT    Maximum longitude of the area
c   LAT        Real(size) OUTPUT   obs latitude
c   LON        Real(size) OUTPUT   obs longitude
c   KFINAL     INTEGER   OUTPUT   Number of obs
c   OBS        Real(size) OUTPUT   Observed parameter value
c   ISTAT      INTEGER   OUTPUT   Status
c
c COMMON BLOCKS: N/A
c

```

```

c FILES: None
c
c DATA BASES: ISIS LLT database
c
c   Name      Table      Usage      Description
c -----
c raob_qc     RAOB_QC      IN        raob obs
c
c NON-FILE INPUT/OUTPUT: N/A
c
c ERROR CONDITIONS:
c   CONDITION      ACTION
c -----
c DTG error        Print err message & return
c Error return from LRD   Print err message
c Error return from LCLOS  Print err message
c
c.....MAINTENANCE SECTION.....
c
c MODULES CALLED:
c   Name      Description
c -----
c LCLOS      ISIS LLT close
c LEN_TRIM   Determines the length of a string
c LRD        ISIS LLT read
c TRIM       Removes the trailing blanks
c
c LOCAL VARIABLES AND      Structures are documented in detail
c   STRUCTURES:      where they are defined in the code
c                   within include files.
c
c METHOD:
c 1. Set seq_type to 'raob_qc'
c
c 2. Get hr from dtg, set mindtg, maxdtg, minhr, maxhr, rsn_in
c   (reporting source name), fcst_in (desired tau), minuptm
c   (min update time), bufflag (0)
c
c 3. Set i=0
c
c   do while (status == 0)
c     call LRD(seq_type, vrsnam, dsetnam, seclvl, mindtg,
c       minhr, maxdtg, maxhr, minlat, maxlat,
c       minlon, maxlon, rsn_in, fcst_in, minuptm,
c       bufflag, rpt_dtg, rpt_hr, rpt_lat, rpt_lon,
c       rpt_rsn, rpt_fcst, rpt_cretm, rpt_udt, llt_id,
c       blkseqid, recseqid, ibuff, fbuff, status)
c   if status /= 0 then
c     if status /= 100 then
c       write error msg
c     end if
c   else

```

```

c      i = i + 1
c      if duplicate data using rpt_rsn
c          print msg
c      endif
c
c      =====
c      pick out the relevant info and fill the array
c      =====
c
c      nobs = 0
c      levels = fbuff % prof_cnt
c      do j=1, levels
c
c          =====
c          pick out only the right level
c          =====
c
c          if (prof_t % pres == level) then
c
c              lat(j) = fbuff % crse_lat
c              lon(j) = fbuff % crse_lon
c              if parm = 'air_temp' then
c                  if prof_t % air_temp /= missing_value then
c                      obs(j) = prof_t % air_temp
c                      nobs = nobs + 1
c                  endif
c              elseif parm = 'geop_ht' then
c                  if prof_t % geop_ht /= missing_value then
c                      obs(j) = prof_t % geop_ht
c                      nobs = nobs + 1
c                  endif
c              elseif parm = 'wnd_dir' then
c                  if prof_t % wnd_dir /= missing_value then
c                      obs(j) = prof_t % wnd_dir
c                      nobs = nobs + 1
c                  endif
c              elseif parm = 'wnd_spd' then
c                  if prof_t % wnd_spd /= missing_value then
c                      obs(j) = prof_t % wnd_spd
c                      nobs = nobs + 1
c                  endif
c              endif
c          endif (right level)
c      enddo (j loop)
c
c      endif c if status=0
c      enddo c i loop
c
c      if (status = 100) then status is good
c      close the dataset by calling LCLOS
c
c      INCLUDE FILES:
c      Name          Description
c      -----

```

```

c  RAOB_QC.H      raob header file
c  common.inc     LLT data structure
c  v_data.h       common variables for verobs
c
c  COMPILER DEPENDENCIES: f90
c
c  COMPILE OPTIONS: -f fixed -c
c
c  MAKEFILE: Located at /a/ops/app/mverif/src/sub/makeverobslib
c      UNICOS make
c
c  RECORD OF CHANGES:
c
c  <<CHANGE NOTICE>> Version 1.1 (29 Apr 1998) -- Kyongsuk Pace
c      Initial submission
c
c.....END PROLOGUE.....
c
      implicit none

      include 'v_data.h'
      include 'common.inc'
      include 'RAOB_QC.H'

c*****
c  formal parameters
c*****
      character(8), intent(in) :: vrsnam
      character(24), intent(in) :: dsetnm
      character(8), intent(in) :: seclvl
      character(10), intent(in) :: dtg
      character(32), intent(in) :: param
      real,          intent(in) :: lvl
      real,          intent(in) :: minlat
      real,          intent(in) :: maxlat
      real,          intent(in) :: minlon
      real,          intent(in) :: maxlon
      real           :: lat(size)
      real           :: lon(size)
      integer        :: kfinal
      real           :: obs(size)
      integer        :: istat

c*****
c  local variables used as arguments for LRD:
c*****
      character(24) :: seq_type
      real          :: hr
      CHARACTER(16) :: MINDTG  ! Minmum date and time group to read.
      CHARACTER(16) :: MAXDTG  ! Maximum date and time group to read.
      REAL          :: MINHR   ! Minmum hour to read.
      REAL          :: MAXHR   ! Maximum hour to read.

```

```

CHARACTER(16) :: RSN_IN   ! Reporting source name.
REAL          :: FCST_IN  ! Desired forecast "TAU".
CHARACTER(24) :: MINUPTM  ! Minimum update time.
CHARACTER(56) :: REMARKS  ! Description of data/assoc. record.
CHARACTER(16) :: RPT_DTG  ! Actual date & time group for report.
REAL          :: RPT_HR   ! Reported hour read.
REAL          :: RPT_LAT  ! Reported latitude read.
REAL          :: RPT_LON  ! Reported longitude read.
CHARACTER(16) :: RPT_RSN  ! Reported longitude read.
REAL          :: RPT_FCST ! Reported "TAU" or forecast time.
CHARACTER(24) :: RPT_CRETM ! Record creation date.
CHARACTER(24) :: RPT_UDT  ! Report's last update time.
INTEGER       :: BUFLAG   ! 0 => Input is in FBUFF
                  ! 1 => Input is in IBUFF
INTEGER       :: LLT_ID   ! Unique database LLT identifier for each
                  ! dataset.
INTEGER       :: BLKSEQID ! Unique database LLT block identifier.
INTEGER       :: RECSEQID ! Unique database LLT record identifier.

```

```

TYPE(raob_qc_int) :: IBUFF
TYPE(raob_qc)     :: FBUFF   ! Real record structure.

```

```

c*****

```

```

c  Arguments for LCLOS (that get "*" for values).

```

```

c*****

```

```

CHARACTER(24) :: SEQTYPE_X ! Report type.
CHARACTER(8)  :: VRSNNAM_X ! Version of ISIS software used.
CHARACTER(24) :: DSETNAM_X ! Data set name used.
CHARACTER(8)  :: SECLVL_X  ! 7 character security
                  ! classification level.
CHARACTER(16) :: DTG_X     ! Date Time Group for write.

```

```

c*****

```

```

c  Other local variables

```

```

c*****

```

```

integer :: levels, lvl_1, nmatch, k
integer :: status, i, j, status2

```

```

seq_type = 'raob_qc' ! obs report type
istat = 0

```

```

c*****

```

```

c  Set up date and time group in YYYYMMDDHH format in DTG.

```

```

c*****

```

```

IF ( LEN_TRIM(DTG) == 10 ) THEN
  READ (UNIT=DTG(9:10),FMT='(F2.0)',IOSTAT=STATUS) HR
  IF ( STATUS == 0 ) THEN
    IF ( HR < 12. ) THEN
      HR = 0.
    ELSE
      HR = 12.
    END IF
  END IF

```

```

ELSE
  WRITE *, ' Cannot read hour ', DTG(9:10),
2    ' from date & time group ', TRIM(DTG)
  istat = -1
  RETURN
END IF
ELSE
  STATUS = 10
  WRITE *, 'raob_qc: Got date and time group ', TRIM(DTG),
2    'of length ',
3    LEN_TRIM(DTG), ' but expected length == 10.'
  istat = -1
  RETURN
END IF

c*****
c  Set the input parameters used to get a read-back value.
c*****
MINDTG = DTG;    MAXDTG = DTG
MINHR  = HR;     MAXHR  = HR+11.999
RSN_IN = '*'
FCST_IN = 0.0 ! Report forecast period or Tau (normal = 0.0)
MINUPTM = '*'
BUFFLAG = 0 ! Want (both) floating (and integer).

c*****
c  get the data from LLT db
c*****
I = 0
kfinal = 0 ! overall number of matched pressure level and param

DO WHILE ( STATUS == 0 )
  CALL LRD(seq_type, vrsnam, dsetnm, SECLVL,
2    MINDTG, MINHR, MAXDTG, MAXHR,
3    MINLAT, MAXLAT, MINLON, MAXLON,
4    RSN_IN, FCST_IN, MINUPTM, BUFFLAG,
5    RPT_DTG, RPT_HR, RPT_LAT, RPT_LON,
6    RPT_RSN, RPT_FCST, RPT_CRETM, RPT_UDT,
7    LLT_ID, BLKSEQID, RECSEQID,
8    IBUFF, FBUFF, STATUS )
  IF ( STATUS /= 0 ) THEN
    IF ( STATUS /= 100 ) THEN
      ! Ignore normal no-more-data return code
      WRITE *, ' Read from ISIS failed. Code = ', STATUS, ' '
      istat = -1
    END IF
    ELSE ! successful LRD
      I = I + 1
  END IF

c*****
c  pick out the relevant info and fill the array
c  prof is a sub-structure of raob_qc

```

```

c      prof_cnt has the number of levels
c*****
      levels = fbuff % prof_cnt

      do j = 1, levels

c*****
c      pick out only the right level
c*****

      nmatch = 0 ! number of matched pressure level and parameter

      lvl_1 = int(fbuff % prof(j) % pres / 100)
      if (lvl_1 == lvl) then

c*****
c      we want to use the qc flag to discard the bad obs
c      use the qc flag value of 1 for this obs type
c*****
      if (param == 'air_temp') then
        if (fbuff%prof(j)%air_temp_qc_id == 1 .and.
2          fbuff%prof(j)%air_temp < check_val) then

c*****
c      want to fill the lat, lon, obs arrays filled from
c      1 to overall number of matched obs without any
c      skipped indices
c      j index is for the level_cnt for a given lat/lon,*
c      k index is for the matched param and pressure level
c      within the j index
c*****
          nmatch = nmatch + 1
          do k = kfinal+1, kfinal+nmatch
            lat(k) = fbuff % crse_lat
            lon(k) = fbuff % crse_lon
            obs(k) = (fbuff % prof(j) % air_temp)
          end do
        end if

      else if (param == 'geop_ht') then
        if (fbuff%prof(j)%geop_ht_qc_id == 1 .and.
2          fbuff%prof(j)%geop_ht < check_val) then
          nmatch = nmatch + 1
          do k = kfinal+1, kfinal+nmatch
            lat(k) = fbuff % crse_lat
            lon(k) = fbuff % crse_lon
            obs(k) = fbuff % prof(j) % geop_ht
          end do
        end if

      else if (param == 'wnd_dir') then
        if (fbuff%prof(j)%wnd_qc_id == 1 .and.

```

```

2      fbuff%prof(j)%dir < check_val) then
      nmatch = nmatch + 1
      do k = kfinal+1, kfinal+nmatch
        lat(k) = fbuff % crse_lat
        lon(k) = fbuff % crse_lon
        obs(k) = fbuff % prof(j) % dir
      end do
    end if

    else if (param == 'wnd_spd') then
      if (fbuff%prof(j)%wnd_qc_id == 1 .and.
2      fbuff%prof(j)%spd < check_val) then
        nmatch = nmatch + 1
        do k = kfinal+1, kfinal+nmatch
          lat(k) = fbuff % crse_lat
          lon(k) = fbuff % crse_lon
          obs(k) = fbuff % prof(j) % spd
        end do
      end if

      end if ! param
    end if ! right level
    kfinal = kfinal + nmatch
  end do ! levels loop
end if ! OK status
end do ! i loop
if (status == 100) status = 0
write *, ''
write *, ' Called LRD ', I, ' times.'
write *, ' Read ', kfinal, ' obs of sequence type ',
2  TRIM(seq_type), ''

c*****
c  Close the dataset (now open for reading) again.
c*****
SEQTYPE_X = '*'; VRSNNAM_X = '*'; DSETNAM_X = '*'
SECLVL_X = '*'; DTG_X = '*'

CALL LCLOS(SEQTYPE_X, VRSNNAM_X, DSETNAM_X,
2  SECLVL_X, DTG_X, STATUS2)
if (STATUS2 /= 0) then
  write *, ' Could not close ISIS table. ',
2  ' Error code is ', STATUS2, ''
end if

return
end subroutine raob_qc_read

```

12. sfcland.f90

```

subroutine sfc_ind_read(vrsnnam, dsetnm, seclvl, dtg, param,

```

```

2          lvl,  minlat, maxlat, minlon, maxlon,
3          lat,  lon,  nobs, obs,  istat)
C
C.....START PROLOGUE.....
C
C SCCS IDENTIFICATION: @(#)sfcland.f90 1.1 04/24/98 /h/cm/library/mverif/src/sub/sfcland.f90_v
C
C CONFIGURATION IDENTIFICATION: NONE
C
C MODULE NAME: sfc_lnd_read
C
C DESCRIPTION: subroutine to read the sfc_land data and pick out the
C              obs data for the given parameter
C
C COPYRIGHT:      (c) 1996 FLENUMMETOCCEN
C                U.S. GOVERNMENT DOMAIN
C                ALL RIGHTS RESERVED
C
C CONTRACT NUMBER AND TITLE: N/A
C
C REFERENCES: NONE
C
C CLASSIFICATION: Unclassified
C
C RESTRICTIONS: NONE
C
C COMPUTER/OPERATING SYSTEM
C              DEPENDENCIES: Cray UNICOS
C
C LIBRARIES OF RESIDENCE: /a/ops/bin
C
C USAGE:
C   call sfc_land(vrsnnam, dsetnm, seclvl, dtg, param,
C               lvl, minlat, maxlat, minlon, maxlon,
C               lat, lon, nobs, obs, istat)
C
C PARAMETERS:
C   Name      Type      Usage      Description
C   -----
C   VRSNNAM    CHAR*8    INPUT    llt version name
C   DSETNM     CHAR*24    INPUT    data set name
C   SECLVL     CHAR*8     INPUT    classification
C   DTG        CHAR*10    INPUT    date time group for read
C   PARAM      CHAR*32    INPUT    parameter
C   LVL        REAL       INPUT    level type
C   MINLAT     REAL       INPUT    minimum latitude
C   MAXLAT     REAL       INPUT    maximum latitude
C   MINLON     REAL       INPUT    minimum longitude
C   MAXLON     REAL       INPUT    maximum longitude
C   LAT        REAL(size) OUTPUT    obs latitude
C   LON        REAL(size) OUTPUT    obs longitude
C   NOBS       INTEGER    OUTPUT    number of obs

```

```

C OBS      REAL(size)  OUTPUT  obs value
C ISTAT    INTEGER     OUTPUT  status code
C
C COMMON BLOCKS: N/A
C
C FILES: None
C
C DATA BASES: ISIS LLT_DB
C   Name      Table  Usage      Description
C   -----
C   sfc_lnd    SFC_LND   IN       surface land obs
C
C NON-FILE INPUT/OUTPUT: N/A
C
C ERROR CONDITIONS:
C   CONDITION      ACTION
C   -----
C   DTG error      Print err message & return
C   Error return from LRD   Print err message
C   Error return from LCLOS  Print err message
C
C ADDITIONAL COMMENTS: NONE
C
C.....MAINTENANCE SECTION.....
C
C MODULES CALLED:
C   Name      Description
C   -----
C   LCLOS     ISIS LLT close
C   LEN_TRIM  Determines the length of a string
C   LRD       ISIS LLT read
C   TRIM      Removes the trailing blanks
C
C LOCAL VARIABLES AND Structures are documented in detail
C   STRUCTURES:   where they are defined in the code
C                 within include files.
C
C METHOD:
C   Set seq_type to 'sfc_lnd'
C   See raob_qc.f90 for the rest.
C
C INCLUDE FILES:
C   Name      Description
C   -----
C   SFC_LND.H  surface land header file
C   V_DATA.H   common variables for verobs
C
C COMPILER DEPENDENCIES: f90
C
C COMPILE OPTIONS: -f fixed -c
C
C MAKEFILE: Located at /a/ops/app/mverif/src/sub/makeverobslib

```

```

C      UNICOS make
C
C RECORD OF CHANGES:
C
C <<CHANGE NOTICE>> Version 1.1 (29 Apr 1998) -- Kyongsuk Pace
C   Initial submission
C
C.....END PROLOGUE.....
C
      implicit none
      include 'v_data.h'
      include 'SFC_LND.H'

```

```

c*****

```

```

c  formal parameters

```

```

c*****

```

```

      character(8), intent(in) :: vrsnam
      character(24), intent(in) :: dsetnm
      character(8), intent(in) :: seclvl
      character(10), intent(in) :: dtg
      character(32), intent(in) :: param
      real,          intent(in) :: lvl
      real,          intent(in) :: minlat
      real,          intent(in) :: maxlat
      real,          intent(in) :: minlon
      real,          intent(in) :: maxlon
      real           :: lat(size)
      real           :: lon(size)
      integer        :: nobs
      real           :: obs(size)
      integer        :: istat

```

```

c*****

```

```

c  local variables used as arguments for LRD:

```

```

c*****

```

```

      character(24) :: seq_type
      real          :: hr
      CHARACTER(16) :: MINDTG  ! Minmum date and time group to read.
      CHARACTER(16) :: MAXDTG  ! Maximum date and time group to read.
      REAL          :: MINHR    ! Minmum hour to read.
      REAL          :: MAXHR    ! Maximum hour to read.
      CHARACTER(16) :: RSN_IN   ! Reporting source name.
      REAL          :: FCST_IN  ! Desired forecast "TAU".
      CHARACTER(24) :: MINUPTM  ! Minimum update time.
      CHARACTER(56) :: REMARKS  ! Description of data/assoc. record.
      CHARACTER(16) :: RPT_DTG  ! Actual date & time group for report.
      REAL          :: RPT_HR    ! Reported hour read.
      REAL          :: RPT_LAT   ! Reported latitude read.
      REAL          :: RPT_LON   ! Reported longitude read.
      CHARACTER(16) :: RPT_RSN   ! Reported longitude read.
      REAL          :: RPT_FCST  ! Reported "TAU" or forecast time.
      CHARACTER(24) :: RPT_CRET  ! Record creation date.

```

```

CHARACTER(24) :: RPT_UDT ! Report's last update time.
INTEGER :: BUFLAG ! 0 => Input is in FBUFF
! 1 => Input is in IBUFF
INTEGER :: LLT_ID ! Unique database LLT identifier for each
! dataset.
INTEGER :: BLKSEQID ! Unique database LLT block identifier.
INTEGER :: RECSEQID ! Unique database LLT record identifier.

```

```

TYPE(sfc_lnd_int) :: IBUFF
TYPE(sfc_lnd) :: FBUFF ! Real record structure.

```

```

c*****

```

```

c Arguments for LCLOS (that get "*" for values).

```

```

c*****

```

```

CHARACTER(24) :: SEQTYPE_X ! Report type.
CHARACTER(8) :: VRSNNAM_X ! Version of ISIS software used.
CHARACTER(24) :: DSETNAM_X ! Data set name used.
CHARACTER(8) :: SECLVL_X ! 7 character security
! classification level.
CHARACTER(16) :: DTG_X ! Date Time Group for write.

```

```

c*****

```

```

c Other local variables

```

```

c*****

```

```

integer :: levels
integer :: status, i, status2

```

```

seq_type = 'sfc_lnd' ! Report type
istat = 0

```

```

|*****

```

```

! Set up date and time group in YYYYMMDDHH format in DTG.

```

```

|*****

```

```

IF ( LEN_TRIM(DTG) == 10 ) THEN
  READ (UNIT=DTG(9:10),FMT='(F2.0)',IOSTAT=STATUS) HR
  IF ( STATUS == 0 ) THEN
    IF ( HR < 12. ) THEN
      HR = 0.
    ELSE
      HR = 12.
    END IF
  ELSE
    WRITE *, ' Cannot read hour "', DTG(9:10),
2      ' from date & time group ', TRIM(DTG)
    istat = -1
    RETURN
  END IF
ELSE
  STATUS = 10
  WRITE *, 'sfcland: Got date and time group ',
2      TRIM(DTG), ' of length ',
3      LEN_TRIM(DTG), ' but expected length == 10.'

```

```

        istat = -1
        RETURN
    END IF

!*****
! Set the input parameters used to get a read-back value.
!*****
    MINDTG = DTG;    MAXDTG = DTG
    MINHR = HR;    MAXHR = HR+11.999
    RSN_IN = '*'
    FCST_IN = 0.0 ! Report forecast period or Tau (normal = 0.0)
    MINUPTM = '*'
    BUFLAG = 0 ! Want (both) floating (and integer).

!*****
! get the data from LLT db
!*****
    I = 0
    nobs = 0

    DO WHILE ( STATUS == 0 )
        CALL LRD(seq_type, vrsnnam, dsetnm, SECLVL,
2         MINDTG, MINHR, MAXDTG, MAXHR,
3         MINLAT, MAXLAT, MINLON, MAXLON,
4         RSN_IN, FCST_IN, MINUPTM, BUFLAG,
5         RPT_DTG, RPT_HR, RPT_LAT, RPT_LON,
6         RPT_RSN, RPT_FCST, RPT_CRETM, RPT_UDT,
7         LLT_ID, BLKSEQID, RECSEQID,
8         Ibuff, Fbuff, STATUS )
        IF ( STATUS /= 0 ) THEN
            IF ( STATUS /= 100 ) THEN
                ! Ignore normal no-more-data return code
                WRITE *, 'Read from ISIS failed. Code = ', STATUS, ' '
                istat = status
            END IF
            ELSE ! successful LRD
                I = I + 1
        END IF

c*****
c pick out the relevant info and fill the array
c check for ISIS missing value
c*****
        if (param == 'air_temp') then
            if (fbuff % air_temp < check_val .and.
2         fbuff % air_temp_qc_id == 1) then
                nobs = nobs + 1
                lat(nobs) = fbuff % crse_lat
                lon(nobs) = fbuff % crse_lon
                obs(nobs) = fbuff % air_temp
            end if

            else if (param == 'wnd_dir') then

```

```

        if (fbuff% wnd_dir < check_val .and.
2         fbuff% wnd_qc_id == 1) then
            nobs = nobs + 1
            lat(nobs) = fbuff% crse_lat
            lon(nobs) = fbuff% crse_lon
            obs(nobs) = fbuff% wnd_dir
        end if

        else if (param == 'sea_lvl_pres') then
            if (fbuff% sea_lvl_pres < check_val .and.
2             fbuff% sea_lvl_pres_qc_id == 1) then
                nobs = nobs + 1
                lat(nobs) = fbuff% crse_lat
                lon(nobs) = fbuff% crse_lon
                obs(nobs) = (fbuff% sea_lvl_pres) / 100.0
            end if

        else if (param == 'wnd_spd') then
            if (fbuff% wnd_spd < check_val .and.
2             fbuff% wnd_qc_id == 1) then
                nobs = nobs + 1
                lat(nobs) = fbuff% crse_lat
                lon(nobs) = fbuff% crse_lon
                obs(nobs) = fbuff% wnd_spd
            end if

        end if ! param
    end if ! OK status
END DO ! i loop

IF ( STATUS == 100 ) STATUS = 0

write *, ''
WRITE *, ' Called LRD ', I, ' times.'
write *, ' Read ', nobs, ' obs of sequence type ', TRIM(SEQ_TYPE),
2     ''

!*****
! Close the dataset (now open for reading) again.
!*****
SEQTYPE_X = '*'; VRSNNAM_X = '*'; DSETNAM_X = '*'
SECLVL_X = '*'; DTG_X = '*'

CALL LCLOS(SEQTYPE_X, VRSNNAM_X, DSETNAM_X,
2     SECLVL_X, DTG_X, STATUS2)
IF ( STATUS2 /= 0 ) THEN
    WRITE *, ' Could not close ISIS table. ',
2     ' Error code is ', STATUS2, ''
END IF

return
end subroutine sfc_lnd_read

```

13. sfcship.f90

```
subroutine sfc_ship_read(vrsnnam, dsetnm, seclvl, dtg, param,
2      lvl, minlat, maxlat, minlon, maxlon,
3      lat, lon, nob, obs, istat)
C
C.....START PROLOGUE.....
C
C SCCS IDENTIFICATION: @(#)sfcship.f90 1.1 04/24/98 /h/cm/library/mverif/src/sub/sfcship.f90_v
C
C CONFIGURATION IDENTIFICATION: NONE
C
C MODULE NAME: sfc_ship_read
C
C DESCRIPTION: subroutine to read the sfc_ship data and pick
C               out the obs data for the given parameter
C
C COPYRIGHT:      (c) 1996 FLENUMMETOCCEN
C                 U.S. GOVERNMENT DOMAIN
C                 ALL RIGHTS RESERVED
C
C CONTRACT NUMBER AND TITLE: N/A
C
C REFERENCES: NONE
C
C CLASSIFICATION: Unclassified
C
C RESTRICTIONS: NONE
C
C COMPUTER/OPERATING SYSTEM
C               DEPENDENCIES: Cray UNICOS
C
C LIBRARIES OF RESIDENCE: /a/ops/bin
C
C USAGE:
C   call sfc_ship_read(vrsnnam, dsetnm, seclvl, dtg, param,
C                     lvl, minlat, maxlat, minlon, maxlon,
C                     lat, lon, nob, obs, istat)
C
C PARAMETERS:
C   Name      Type      Usage      Description
C   -----
C   VRSNNAM    CHAR*8    INPUT    llt version name
C   DSETNM     CHAR*24    INPUT    data set name
C   SECLVL     CHAR*8     INPUT    classification
C   DTG        CHAR*10    INPUT    date time group for read
C   PARAM      CHAR*32    INPUT    parameter
C   LVL        REAL       INPUT    level type
C   MINLAT     REAL       INPUT    minimum latitude
C   MAXLAT     REAL       INPUT    maximum latitude
C   MINLON     REAL       INPUT    minimum longitude
```

```

C MAXLON      REAL      INPUT  maximum longitude
C LAT         REAL(size) OUTPUT obs latitude
C LON         REAL(size) OUTPUT obs longitude
C NOBS        INTEGER    OUTPUT number of obs
C OBS         REAL(size) OUTPUT obs value
C ISTAT       INTEGER    OUTPUT return status
C
C COMMON BLOCKS: N/A
C
C FILES: None
C
C DATA BASES: ISIS LLT_DB
C   Name      Table      Usage      Description
C   -----
C   sfc_ship   SFC_SHIP    IN       surface ship obs
C
C NON-FILE INPUT/OUTPUT: N/A
C
C ERROR CONDITIONS:
C   CONDITION      ACTION
C   -----
C   DTG error      Print err message & return
C   Error return from LRD   Print err message
C   Error return from LCLOS Print err message
C
C ADDITIONAL COMMENTS: NONE
C
C.....MAINTENANCE SECTION.....
C
C MODULES CALLED:
C   Name      Description
C   -----
C   LCLOS     ISIS LLT close
C   LEN_TRIM  Determines the length of a string
C   LRD       ISIS LLT read
C   TRIM      Removes the trailing blanks
C
C LOCAL VARIABLES AND Structures are documented in detail
C   STRUCTURES:   where they are defined in the code
C                 within include files.
C
C METHOD:
C   Set seq_type to 'sfc_ship'
C   See raob_qc_read for the rest.
C
C INCLUDE FILES:
C   Name      Description
C   -----
C   SFC_SHIP.H  surface ship header file
C   V_DATA.H    common variables for verobs
C
C COMPILER DEPENDENCIES: f90

```

```

C
C COMPILE OPTIONS: -f fixed -c
C
C MAKEFILE: Located at /a/ops/app/mverif/src/sub/makeverobslib
C      UNICOS make
C
C RECORD OF CHANGES:
C
C <<CHANGE NOTICE>> Version 1.1 (29 Apr 1998) -- Kyongsuk Pace
C      Initial submission
C
C.....END PROLOGUE.....
C
      implicit none
      include 'v_data.h'
      include 'SFC_SHIP.H'

```

```

c*****

```

```

c  formal parameters

```

```

c*****

```

```

      character(8), intent(in) :: vrsnnam
      character(24), intent(in) :: dsetnm
      character(8), intent(in) :: seclvl
      character(10), intent(in) :: dtg
      character(32), intent(in) :: param
      real,          intent(in) :: lvl
      real,          intent(in) :: minlat
      real,          intent(in) :: maxlat
      real,          intent(in) :: minlon
      real,          intent(in) :: maxlon
      real           :: lat(size)
      real           :: lon(size)
      integer        :: nob
      real           :: obs(size)
      integer        :: istat

```

```

c*****

```

```

c  local variables used as arguments for LRD:

```

```

c*****

```

```

      character(24) :: seq_type
      real          :: hr
      CHARACTER(16) :: MINDTG  ! Minmum date and time group to read.
      CHARACTER(16) :: MAXDTG  ! Maximum date and time group to read.
      REAL          :: MINHR   ! Minmum hour to read.
      REAL          :: MAXHR   ! Maximum hour to read.
      CHARACTER(16) :: RSN_IN  ! Reporting source name.
      REAL          :: FCST_IN ! Desired forecast "TAU".
      CHARACTER(24) :: MINUPTM ! Minimum update time.
      CHARACTER(56) :: REMARKS ! Description of data/assoc. record.
      CHARACTER(16) :: RPT_DTG ! Actual date & time group for report.
      REAL          :: RPT_HR  ! Reported hour read.
      REAL          :: RPT_LAT ! Reported latitude read.

```

```

REAL      :: RPT_LON  ! Reported longitude read.
CHARACTER(16) :: RPT_RSN ! Reported longitude read.
REAL      :: RPT_FCST ! Reported "TAU" or forecast time.
CHARACTER(24) :: RPT_CRETM ! Record creation date.
CHARACTER(24) :: RPT_UDT ! Report's last update time.
INTEGER     :: BUFLAG  ! 0 => Input is in FBUFF
                ! 1 => Input is in IBUFF
INTEGER     :: LLT_ID  ! Unique database LLT identifier for each
                ! dataset.
INTEGER     :: BLKSEQID ! Unique database LLT block identifier.
INTEGER     :: RECSEQID ! Unique database LLT record identifier.

TYPE(sfc_ship_int):: IBUFF ! integer record structure.
TYPE(sfc_ship):: FBUFF  ! Real record structure.

c*****
c Arguments for LCLOS (that get "*" for values).
c*****
CHARACTER(24) :: SEQTYPE_X ! Report type.
CHARACTER(8)  :: VRSNNAM_X ! Version of ISIS software used.
CHARACTER(24) :: DSETNAM_X ! Data set name used.
CHARACTER(8)  :: SECLVL_X  ! 7 character security
                ! classification level.
CHARACTER(16) :: DTG_X     ! Date Time Group for write.

c*****
c Other local variables
c*****
integer :: levels
integer :: status, i, status2

seq_type = 'sfc_ship' ! Report type
istat = 0

!*****
! Set up date and time group in YYYYMMDDHH format in DTG.
!*****
IF ( LEN_TRIM(DTG) == 10 ) THEN
  READ (UNIT=DTG(9:10),FMT='(F2.0)',IOSTAT=STATUS) HR
  IF ( STATUS == 0 ) THEN
    IF ( HR < 12. ) THEN
      HR = 0.
    ELSE
      HR = 12.
    END IF
  ELSE
    WRITE *, ' Cannot read hour "', DTG(9:10),
2      ' from date & time group ', TRIM(DTG)
    istat = -1
    RETURN
  END IF
ELSE

```

```

        STATUS = 10
        WRITE *, 'sfcsip: Got date and time group ',
2         TRIM(DTG), ' of length ',
3         LEN_TRIM(DTG), ' but expected length == 10.'
        istat = -1
        RETURN
    END IF

!*****
!   Set the input parameters used to get a read-back value.
!*****
        MINDTG = DTG;   MAXDTG = DTG
        MINHR  = HR;    MAXHR  = HR+11.999
        RSN_IN = '*'
        FCST_IN = 0.0   ! Report forecast period or Tau (normal = 0.0)
        MINUPTM = '*'
        BUFFLAG = 0     ! Want (both) floating (and integer).

!*****
!   get the data from LLT db
!*****
        I = 0
        nobs = 0

        DO WHILE ( STATUS == 0 )
            CALL LRD(seq_type, vrsnam, dsetnm, SECLVL,
2             MINDTG, MINHR, MAXDTG, MAXHR,
3             MINLAT, MAXLAT, MINLON, MAXLON,
4             RSN_IN, FCST_IN, MINUPTM, BUFFLAG,
5             RPT_DTG, RPT_HR, RPT_LAT, RPT_LON,
6             RPT_RSN, RPT_FCST, RPT_CRETM, RPT_UDT,
7             LLT_ID, BLKSEQID, RECSEQID,
8             IBUFF, FBUFF, STATUS )
            IF ( STATUS /= 0 ) THEN
                IF ( STATUS /= 100 ) THEN
                    ! Ignore normal no-more-data return code
                    WRITE *, 'Read from ISIS failed. Code = ', STATUS, '!'
                    istat = status
                END IF
            ELSE ! successful LRD
                I = I + 1
            END IF
        END DO

c*****
c   pick out the relevant info and fill the array
c   we want to use the qc flag to discard the bad obs when available
c*****

        if (param == 'air_temp') then
            if (fbuff% air_temp < check_val) then
                nobs = nobs + 1
                lat(nobs) = fbuff% crse_lat
                lon(nobs) = fbuff% crse_lon
            end if
        end if

```

```

        obs(nobs) = fbuff % air_temp
    end if

    else if (param == 'sea_lvl_pres') then
        if (fbuff % sea_lvl_pres < check_val) then
            nobs = nobs + 1
            lat(nobs) = fbuff % crse_lat
            lon(nobs) = fbuff % crse_lon
            obs(nobs) = (fbuff % sea_lvl_pres) / 100.0
        end if

        else if (param == 'sea_temp') then
            !check pos_qc_id for position error first
            if (fbuff % pos_qc_id == 0 .or.
2         fbuff % pos_qc_id == 1) then
                !sea_temp_qc flag of 0 or 1 is the only obs we want to use
                if ( (fbuff % sea_temp_qc_id == 1 .or.
2         fbuff % sea_temp_qc_id == 0) .or.
3         (fbuff % sea_temp < check_val) ) then
                    nobs = nobs + 1
                    lat(nobs) = fbuff % crse_lat
                    lon(nobs) = fbuff % crse_lon
                    obs(nobs) = fbuff % sea_temp
                end if
            end if

            !for inst_wav_per (grid parm 'peak_wav_per')
            else if (param == 'inst_wav_per') then
                if (fbuff % inst_wav_per < check_val) then
                    nobs = nobs + 1
                    lat(nobs) = fbuff % crse_lat
                    lon(nobs) = fbuff % crse_lon
                    obs(nobs) = fbuff % inst_wav_per
                end if

                !for inst_wav_ht_2 (grid parm 'sig_wav_ht')
                else if (param == 'inst_wav_ht_2') then
                    if (fbuff % inst_wav_ht_2 < check_val) then
                        nobs = nobs + 1
                        lat(nobs) = fbuff % crse_lat
                        lon(nobs) = fbuff % crse_lon
                        obs(nobs) = fbuff % inst_wav_ht_2
                    end if

                    else if (param == 'wnd_dir') then
                        if (fbuff % wnd_dir < check_val) then
                            nobs = nobs + 1
                            lat(nobs) = fbuff % crse_lat
                            lon(nobs) = fbuff % crse_lon
                            obs(nobs) = fbuff % wnd_dir
                        end if

```

```

    else if (param == 'wnd_spd') then
      if (fbuff % wnd_spd < check_val) then
        nob = nob + 1
        lat(nob) = fbuff % crse_lat
        lon(nob) = fbuff % crse_lon
        obs(nob) = fbuff % wnd_spd
      end if

    end if ! param
  end if ! OK status
END DO ! i loop

IF ( STATUS == 100 ) STATUS = 0
write *, ''
WRITE *, ' Called LRD ', I, ' times.'
write *, ' Read ', nob, ' obs of sequence type ', TRIM(SEQ_TYPE),
2      ''

!*****
! Close the dataset (now open for reading) again.
!*****
SEQTYPE_X = '*'; VRSNNAM_X = '*'; DSETNAM_X = '*'
SECLVL_X = '*'; DTG_X = '*'

CALL LCLOS(SEQTYPE_X, VRSNNAM_X, DSETNAM_X,
2      SECLVL_X, DTG_X, STATUS2)
IF ( STATUS2 /= 0 ) THEN
  WRITE *, ' Could not close ISIS table. ',
2      ' Error code is ', STATUS2, ''
END IF

return
end subroutine sfc_ship_read

```

14. ssmetqc.f90

```

subroutine sfc_ship_met_qc_read(vrsnnam, dsetnm, seclvl, dtg,
2      param, lvl, minlat, maxlat,
3      minlon, maxlon, lat, lon,
4      nob, obs, istat)
C
C.....START PROLOGUE.....
C
C SCCS IDENTIFICATION:  @(#)ssmetqc.f90 1.1 04/24/98 /h/cm/library/mverif/src/sub/ssmetqc.f90_v
C
C CONFIGURATION IDENTIFICATION:  NONE
C
C MODULE NAME:  sfc_ship_met_qc_read
C
C DESCRIPTION:  subroutine to read the sfc_ship_met_qc data and pick
C               out the obs data for the given parameter

```

```

C
C COPYRIGHT:          (c) 1996 FLENUMMETOCEN
C                   U.S. GOVERNMENT DOMAIN
C                   ALL RIGHTS RESERVED
C
C CONTRACT NUMBER AND TITLE:  N/A
C
C REFERENCES: NONE
C
C CLASSIFICATION:  Unclassified
C
C RESTRICTIONS: NONE
C
C COMPUTER/OPERATING SYSTEM
C       DEPENDENCIES: Cray UNICOS
C
C LIBRARIES OF RESIDENCE:  /a/ops/bin
C
C USAGE:
C   call sfc_ship_met_qc_read(vrsnnam, dsetnm, seclvl, dtg,
C                             param, lvl, minlat, maxlat,
C                             minlon, maxlon, lat, lon,
C                             nob, obs, istat)
C
C PARAMETERS:
C   Name      Type      Usage      Description
C   -----
C   VRSNNAM    CHAR*8    INPUT    llt version name
C   DSETNM     CHAR*24    INPUT    data set name
C   SECLVL     CHAR*8     INPUT    classification
C   DTG        CHAR*10    INPUT    date time group for read
C   PARAM      CHAR*32    INPUT    parameter
C   LVL        REAL       INPUT    level type
C   MINLAT     REAL       INPUT    minimum latitude
C   MAXLAT     REAL       INPUT    maximum latitude
C   MINLON     REAL       INPUT    minimum longitude
C   MAXLON     REAL       INPUT    maximum longitude
C   LAT        REAL(size) OUTPUT    obs latitude
C   LON        REAL(size) OUTPUT    obs longitude
C   NOBS       INTEGER    OUTPUT    number of obs
C   OBS        REAL(size) OUTPUT    obs value
C   ISTAT      INTEGER    OUTPUT    return status
C
C COMMON BLOCKS: N/A
C
C FILES: None
C
C DATA BASES: ISIS LLT_DB
C   Name      Table      Usage      Description
C   -----
C   sfc_ship_met_qc SFC_SHIP_MET_QC  IN      surface ship met qc obs
C

```

```

C NON-FILE INPUT/OUTPUT: N/A
C
C ERROR CONDITIONS:
C   CONDITION          ACTION
C   -----
C   DTG error          Print err message & return
C   Error return from LRD   Print err message
C   Error return from LCLOS  Print err message
C
C ADDITIONAL COMMENTS: NONE
C
C.....MAINTENANCE SECTION.....
C
C MODULES CALLED:
C   Name      Description
C   -----
C   LCLOS     ISIS LLT close
C   LEN_TRIM   Determines the length of a string
C   LRD       ISIS LLT read
C   TRIM       Removes the trailing blanks
C
C LOCAL VARIABLES AND   Structures are documented in detail
C   STRUCTURES:         where they are defined in the code
C                       within include files.
C
C METHOD:
C   Set seq_type to 'sfc_ship_met_qc'
C   See raob_qc_read for the rest.
C
C INCLUDE FILES:
C   Name      Description
C   -----
C   SFC_SHIP_MET_QC.H  surface ship header file
C   V_DATA.H          common variables for verobs
C
C COMPILER DEPENDENCIES: f90
C
C COMPILE OPTIONS: -f fixed -c
C
C MAKEFILE: Located at /a/ops/app/mverif/src/sub/makeverobslib
C   UNICOS make
C
C RECORD OF CHANGES:
C
C <<CHANGE NOTICE>> Version 1.1 (29 Apr 1998) -- Kyongsuk Pace
C   Initial submission
C
C.....END PROLOGUE.....
C
implicit none
include 'v_data.h'
include 'SFC_SHIP_MET_QC.H'

```

```

c*****
c  formal parameters
c*****
character(8), intent(in) :: vrsnnam
character(24), intent(in) :: dsetnm
character(8), intent(in) :: seclvl
character(10), intent(in) :: dtg
character(32), intent(in) :: param
real,      intent(in) :: lvl
real,      intent(in) :: minlat
real,      intent(in) :: maxlat
real,      intent(in) :: minlon
real,      intent(in) :: maxlon
real       :: lat(size)
real       :: lon(size)
integer    :: nob
real       :: obs(size)
integer    :: istat

c*****
c  local variables used as arguments for LRD:
c*****
character(24) :: seq_type
real          :: hr
CHARACTER(16) :: MINDTG  ! Minnum date and time group to read.
CHARACTER(16) :: MAXDTG  ! Maximum date and time group to read.
REAL         :: MINHR    ! Minnum hour to read.
REAL         :: MAXHR    ! Maximum hour to read.
CHARACTER(16) :: RSN_IN  ! Reporting source name.
REAL         :: FCST_IN  ! Desired forecast "TAU".
CHARACTER(24) :: MINUPTM ! Minimum update time.
CHARACTER(56) :: REMARKS ! Description of data/assoc. record.
CHARACTER(16) :: RPT_DTG ! Actual date & time group for report.
REAL         :: RPT_HR   ! Reported hour read.
REAL         :: RPT_LAT  ! Reported latitude read.
REAL         :: RPT_LON  ! Reported longitude read.
CHARACTER(16) :: RPT_RSN ! Reported longitude read.
REAL         :: RPT_FCST ! Reported "TAU" or forecast time.
CHARACTER(24) :: RPT_CRETM ! Record creation date.
CHARACTER(24) :: RPT_UDT ! Report's last update time.
INTEGER       :: BUFLAG  ! 0 => Input is in FBUFF
                ! 1 => Input is in IBUFF
INTEGER       :: LLT_ID  ! Unique database LLT identifier for each
                ! dataset.
INTEGER       :: BLKSEQID ! Unique database LLT block identifier.
INTEGER       :: RECSEQID ! Unique database LLT record identifier.

TYPE(sfc_ship_met_qc_int) :: IBUFF
TYPE(sfc_ship_met_qc)    :: FBUFF ! Real record structure.

c*****
c  Arguments for LCLOS (that get "*" for values).

```

```

c*****
CHARACTER(24) :: SEQTYPE_X ! Report type.
CHARACTER(8)  :: VRSNNAM_X ! Version of ISIS software used.
CHARACTER(24) :: DSETNAM_X ! Data set name used.
CHARACTER(8)  :: SECLVL_X  ! 7 character security
                  ! classification level.
CHARACTER(16) :: DTG_X    ! Date Time Group for write.

c*****
c  Other local variables
c*****
integer :: levels
integer :: status, i, status2

seq_type = 'sfc_ship_met_qc' ! Report type
istat = 0

!*****
!  Set up date and time group in YYYYMMDDHH format in DTG.
!*****
IF ( LEN_TRIM(DTG) == 10 ) THEN
  READ (UNIT=DTG(9:10),FMT='(F2.0)',IOSTAT=STATUS) HR
  IF ( STATUS == 0 ) THEN
    IF ( HR < 12. ) THEN
      HR = 0.
    ELSE
      HR = 12.
    END IF
  ELSE
    WRITE *, ' Cannot read hour "', DTG(9:10),
2      ' from date & time group ', TRIM(DTG)
    istat = -1
    RETURN
  END IF
ELSE
  STATUS = 10
  WRITE *, 'ssmetqc: Got date and time group ',
2      TRIM(DTG), ' of length ',
3      LEN_TRIM(DTG), ' but expected length == 10.'
  istat = -1
  RETURN
END IF

!*****
!  Set the input parameters used to get a read-back value.
!*****
MINDTG = DTG;    MAXDTG = DTG
MINHR  = HR;     MAXHR  = HR+11.999
RSN_IN = '*'
FCST_IN = 0.0 ! Report forecast period or Tau (normal = 0.0)
MINUPTM = '*'
BUFFLAG = 0    ! Want (both) floating (and integer).

```

```

*****
! get the data from LLT db
*****

I = 0
nobs = 0

DO WHILE ( STATUS == 0 )
  CALL LRD(seq_type, vrsnam, dsetnm, SECLVL,
2    MINDTG, MINHR, MAXDTG, MAXHR,
3    MINLAT, MAXLAT, MINLON, MAXLON,
4    RSN_IN, FCST_IN, MINUPTM, BUFLAG,
5    RPT_DTG, RPT_HR, RPT_LAT, RPT_LON,
6    RPT_RSN, RPT_FCST, RPT_CRETM, RPT_UDT,
7    LLT_ID, BLKSEQID, RECSEQID,
8    IBUFF, FBUFF, STATUS )
  IF ( STATUS /= 0 ) THEN
    IF ( STATUS /= 100 ) THEN
      ! Ignore normal no-more-data return code
      WRITE *, 'Read from ISIS failed. Code = ', STATUS, ' '
      istat = status
    END IF
    ELSE ! successful LRD
      I = I + 1

c*****
c    pick out the relevant info and fill the array
c    we want to use the qc flag to discard the bad obs
c*****
    if (param == 'air_temp') then
      if (fbuff % air_temp < check_val .and.
2      fbuff % air_temp_qc_id == 1) then
        nobs = nobs + 1
        lat(nobs) = fbuff % crse_lat
        lon(nobs) = fbuff % crse_lon
        obs(nobs) = fbuff % air_temp
      end if

    else if (param == 'sea_lvl_pres') then
      if (fbuff % sea_lvl_pres < check_val .and.
2      fbuff % sea_lvl_pres_qc_id == 1) then
        nobs = nobs + 1
        lat(nobs) = fbuff % crse_lat
        lon(nobs) = fbuff % crse_lon
        obs(nobs) = (fbuff % sea_lvl_pres) / 100.0
      end if

    else if (param == 'sea_temp') then
      if (fbuff % sea_temp < check_val .and.
2      fbuff % sea_temp_qc_id == 1) then
        nobs = nobs + 1
        lat(nobs) = fbuff % crse_lat
        lon(nobs) = fbuff % crse_lon

```

```

        obs(nobs) = fbuff % sea_temp
    end if

    else if (param == 'wnd_dir') then
        if (fbuff % wnd_dir < check_val .and.
2         fbuff % wnd_qc_id == 1) then
            nobs = nobs + 1
            lat(nobs) = fbuff % crse_lat
            lon(nobs) = fbuff % crse_lon
            obs(nobs) = fbuff % wnd_dir
        end if

    else if (param == 'wnd_spd') then
        if (fbuff % wnd_spd < check_val .and.
2         fbuff % wnd_qc_id == 1) then
            nobs = nobs + 1
            lat(nobs) = fbuff % crse_lat
            lon(nobs) = fbuff % crse_lon
            obs(nobs) = fbuff % wnd_spd
        end if

    end if ! param
end if ! OK status
END DO ! i loop

IF ( STATUS == 100 ) STATUS = 0
write *, ''
WRITE *, ' Called LRD ', I, ' times.'
write *, ' Read ', nobs, ' obs of sequence type ', TRIM(SEQ_TYPE),
2     ''

!*****
!   Close the dataset (now open for reading) again.
!*****
SEQTYPE_X = '*'; VRSNNAM_X = '*'; DSETNAM_X = '*'
SECLVL_X = '*'; DTG_X = '*'

CALL LCLOS(SEQTYPE_X, VRSNNAM_X, DSETNAM_X,
2     SECLVL_X, DTG_X, STATUS2)
IF ( STATUS2 /= 0 ) THEN
    WRITE *, ' Could not close ISIS table. ',
2     ' Error code is ', STATUS2, ''
END IF

return
end subroutine sfc_ship_met_qc_read

```

B. STAT LIB

1. Compute_bias module

```
SUBROUTINE COMPUTE_BIAS (array1, array2,
2      arr_size, geomname, bias)
C
C.....START PROLOGUE.....
C
C SCCS IDENTIFICATION: @(#)find-bias.f90 1.1 04/24/98 /h/cm/library/mverif/src/sub/find-bias.f90_v
C
C CONFIGURATION IDENTIFICATION: NONE
C
C MODULE NAME: compute_bias
C
C DESCRIPTION: subroutine to compute the bias (mean error)
C
C COPYRIGHT:      (c) 1998 FLENUMMETOCCEN
C                U.S. GOVERNMENT DOMAIN
C                ALL RIGHTS RESERVED
C
C CONTRACT NUMBER AND TITLE: N/A
C
C CLASSIFICATION: Unclassified
C
C RESTRICTIONS: NONE
C
C COMPUTER/OPERATING SYSTEM
C      DEPENDENCIES: Cray UNICOS
C
C LIBRARIES OF RESIDENCE: /a/ops/bin
C
C USAGE:
C   call compute_bias( array1, array2, arr_size, geomname, bias)
C
C PARAMETERS:
C   Name      Type      Usage      Description
C   -----
C   array1    REAL(360*181) INPUT  first array
C   array2    REAL(360*181) INPUT  second array
C   arr_size  INTEGER     INPUT  array size
C   geomname  CHAR*32     INPUT  geometry name
C   bias      REAL        OUTPUT  computed bias
C
C COMMON BLOCKS: N/A
C
C FILES: None
C
C ERROR CONDITIONS:
C   CONDITION      ACTION
```

```

C -----
C
C ADDITIONAL COMMENTS: NONE
C
C.....MAINTENANCE SECTION.....
C
C MODULES CALLED:
C   Name      Description
C   -----
C   FIND_MAP_FACT determine the map factor for the geometry
C
C LOCAL VARIABLES AND      Structures are documented in detail
C   STRUCTURES:      where they are defined in the code
C                   within include files.
C METHOD:
C
C INCLUDE FILES: NONE
C
C COMPILER DEPENDENCIES: f90
C
C COMPILE OPTIONS: -f fixed -c
C
C MAKEFILE: Located at /a/ops/app/mverif/src/sub/makestatlib
C           UNICOS make
C
C RECORD OF CHANGES:
C
C <<CHANGE NOTICE>> Version 1.1 (29 Apr 1998) -- Kyongsuk Pace
C   Initial submission
C
C.....END PROLOGUE.....
C
!*****
! formula used:
! (array1 - array2) / arr_size
! [(array1 - array2) * xmap_factor * ymap_factor]
! / [4*pi*a_square]
!*****
implicit none

integer:: im, jm, imjm
parameter(im = 360)
parameter(jm = 181)
parameter(imjm = im * jm)

!*****
!formal parameter
!*****
real, intent(in)      :: array1(imjm)
real, intent(in)      :: array2(imjm)
integer, intent(in)   :: arr_size
character(32)         :: geomname

```

```

real          :: bias

!*****
!local var
!*****
integer :: i
integer :: lengeom
real    :: sum, dif, sumx
real    :: xmap_factor(imjm), ymap_factor(imjm)
real    :: pi, a_square
integer strlen

pi = 2.0 * asin(1.0)
a_square = (6.375e+06) ** 2
sum = 0.
dif = 0.

if (geomname(1:4) /= 'NONE') then
  write *, "calling find-map-factor from find-bias for ",
2    geomname
  CALL FIND_MAP_FACTOR(geomname, xmap_factor, ymap_factor)
  do i = 1, arr_size
    dif = (array1(i) - array2(i))
2    * xmap_factor(i) * ymap_factor(i)
    sum = sum + dif
  end do
  bias = sum / (4*pi*a_square)
else
  do i = 1, arr_size
    dif = array1(i) - array2(i)
    sum = sum + dif
  end do
  bias = sum / arr_size
end if

return
END SUBROUTINE COMPUTE_BIAS

```

2. Compute_rms

```

SUBROUTINE COMPUTE_RMS (array1, array2,
2    arr_size, geomname, rms)
C
C.....START PROLOGUE.....
C
C SCCS IDENTIFICATION: @(#)find-rms.f90 1.1 04/24/98 /h/cm/library/mverif/src/sub/find-rms.f90_v
C
C CONFIGURATION IDENTIFICATION: NONE
C
C MODULE NAME: compute_rms
C

```

```

C DESCRIPTION: subroutine to compute the rms
C
C COPYRIGHT:      (c) 1998 FLENUMMETOCCEN
C                U.S. GOVERNMENT DOMAIN
C                ALL RIGHTS RESERVED
C
C CONTRACT NUMBER AND TITLE: N/A
C
C CLASSIFICATION: Unclassified
C
C RESTRICTIONS: NONE
C
C COMPUTER/OPERATING SYSTEM
C                DEPENDENCIES: Cray UNICOS
C
C LIBRARIES OF RESIDENCE: /a/ops/bin
C
C USAGE:
C call compute_rms( array1, array2, arr_size, geomname, rms)
C
C PARAMETERS:
C   Name      Type      Usage      Description
C   -----
C array1      REAL(360*181) INPUT  first array
C array2      REAL(360*181) INPUT  second array
C arr_size    INTEGER     INPUT  array size
C geomname    CHAR*32     INPUT  geometry name
C rms         REAL        OUTPUT  computed rms
C
C COMMON BLOCKS: N/A
C
C FILES: None
C
C ERROR CONDITIONS:
C   CONDITION      ACTION
C   -----
C
C ADDITIONAL COMMENTS: NONE
C
C.....MAINTENANCE SECTION.....
C
C MODULES CALLED:
C   Name      Description
C   -----
C FIND_MAP_FACT determine the map factor for the geometry
C
C LOCAL VARIABLES AND Structures are documented in detail
C   STRUCTURES:      where they are defined in the code
C                   within include files.
C
C METHOD:
C
C INCLUDE FILES: NONE

```

```

C
C COMPILER DEPENDENCIES: f90
C
C COMPILE OPTIONS: -f fixed -c
C
C MAKEFILE: Located at /a/ops/app/mverif/src/sub/makestatlib
C          UNICOS make
C
C RECORD OF CHANGES:
C
C <<CHANGE NOTICE>> Version 1.1 (29 Apr 1998) -- Kyongsuk Pace
C   Initial submission
C
C.....END PROLOGUE.....
C
!*****
!sqrt[(array1-array2)**2/arr_size]
!sqrt[(array1-array2)**2*xmap_factor*ymap_factor]
! / [4*pi*a_square]
!*****
implicit none

integer:: im, jm, imjm
parameter(im = 360)
parameter(jm = 181)
parameter(imjm = im * jm)

!*****
!formal parameters
!*****
real,    intent(in) :: array1(imjm)
real,    intent(in) :: array2(imjm)
integer, intent(in) :: arr_size
character(32), intent(in) :: geomname
real      :: rms

!*****
!local var
!*****
integer :: i
real    :: xmap_factor(imjm), ymap_factor(imjm)
real    :: sum, dif, sumx, difx
real    :: pi, a_square

pi = 2.0 * asin(1.0)
a_square = (6.375e+06) ** 2
difx = 0.
sumx = 0.

if (geomname(1:4) /= 'NONE') then
  CALL FIND_MAP_FACTOR(geomname, xmap_factor, ymap_factor)
  do i = 1, arr_size

```

```

      difx = (array1(i) - array2(i))**2
2      * xmap_factor(i) * ymap_factor(i)
      sumx = sumx + difx
    end do
    rms = sqrt(sumx) / (4*pi*a_square)
  else
    do i = 1, arr_size
      difx = (array1(i) - array2(i))**2
      sumx = sumx + difx
    end do
    rms = sqrt(sumx / arr_size)
  end if

  return
END SUBROUTINE COMPUTE_RMS

```

3. Compute_std

```

SUBROUTINE COMPUTE_STD (array1, array2,
2      arr_size, geomname, std)
C
C.....START PROLOGUE.....
C
C SCCS IDENTIFICATION: @(#)find-std.f90 1.1 04/24/98 /h/cm/library/mverif/src/sub/find-std.f90_v
C
C CONFIGURATION IDENTIFICATION: NONE
C
C MODULE NAME: compute_std
C
C DESCRIPTION: subroutine to compute the std
C
C COPYRIGHT:      (c) 1998 FLENUMMETOCCEN
C                U.S. GOVERNMENT DOMAIN
C                ALL RIGHTS RESERVED
C
C CONTRACT NUMBER AND TITLE: N/A
C
C CLASSIFICATION: Unclassified
C
C RESTRICTIONS: NONE
CC COMPUTER/OPERATING SYSTEM
C      DEPENDENCIES: Cray UNICOS
C
C LIBRARIES OF RESIDENCE: /a/ops/bin
C
C USAGE:
C  call compute_std( array1, array2, arr_size, geomname, std)
C
C PARAMETERS:
C  Name      Type      Usage      Description
C  -----

```

```

C array1      REAL(360*181) INPUT  first array
C array2      REAL(360*181) INPUT  second array
C arr_size    INTEGER    INPUT  array size
C geomname    CHAR*32    INPUT  geometry name
C std         REAL        OUTPUT  computed std
C
C COMMON BLOCKS: N/A
C
C FILES: None
C
C ERROR CONDITIONS:
C   CONDITION          ACTION
C   -----
C
C ADDITIONAL COMMENTS: NONE
C
C.....MAINTENANCE SECTION.....
C
C MODULES CALLED:
C   Name      Description
C   -----
C FIND_MAP_FACT determine the map factor for the geometry
C
C LOCAL VARIABLES AND Structures are documented in detail
C   STRUCTURES:   where they are defined in the code
C                 within include files.
C METHOD:
C
C INCLUDE FILES: NONE
C
C COMPILER DEPENDENCIES: f90
C
C COMPILE OPTIONS: -f fixed -c
C
C MAKEFILE: Located at /a/ops/app/mverif/src/sub/makestatlib
C   UNICOS make
C
C RECORD OF CHANGES:
C
C <<CHANGE NOTICE>> Version 1.1 (29 Apr 1998) -- Kyongsuk Pace
C   Initial submission
C
C.....END PROLOGUE.....
C
C *****
C !sqrt[{(array1-array2)**2/arr_size} -
C !  {(array1-array2)/arr_size)**2}]
C !sqrt[{(array1-array2)**2*xmap_factor*ymap_factor}
C !  -{(array1-array2)*xmap_factor*ymap_factor)**2}]
C ! / [4*pi*a_square]
C !*****
C implicit none

```

```

integer:: im, jm, imjm
parameter(im = 360)
parameter(jm = 181)
parameter(imjm = im * jm)

!*****
!formal parameter
!*****
real, intent(in)      :: array1(imjm)
real, intent(in)      :: array2(imjm)
integer, intent(in)   :: arr_size
character(32), intent(in) :: geomname
real                  :: std

!*****
!local var
!*****
integer :: i
real    :: xmap_factor(imjm), ymap_factor(imjm)
real    :: sum, dif, sumx, difx
real    :: pi, a_square

pi = 2.0 * asin(1.0)
a_square = (6.375e+06) ** 2
sum = 0.
dif = 0.
difx = 0.
sumx = 0.

if (geomname(1:4) /= 'NONE') then
  CALL FIND_MAP_FACTOR(geomname, xmap_factor, ymap_factor)
  do i = 1, arr_size
    dif = (array1(i) - array2(i))
2    * xmap_factor(i) * ymap_factor(i)
    difx = (array1(i) - array2(i))**2
2    * xmap_factor(i) * ymap_factor(i)
    sum = sum + dif
    sumx = sumx + difx
  end do
  std = sqrt(sumx / 4*pi*a_square - (sum / (4*pi*a_square))**2)
else
  do i = 1, arr_size
    dif = array1(i) - array2(i)
    sum = sum + dif
    difx = (array1(i) - array2(i))**2
    sumx = sumx + difx
  end do
  std = sqrt(sumx/arr_size - (sum/arr_size)**2)
end if

return

```

END SUBROUTINE COMPUTE_STD

C. GRAPHICS

1. Plot_data.pro

pro plot_data

```
*****
;
; setup color chart
*****
COMMON colors, r_orig, g_orig, b_orig, r_curr, g_curr, b_curr
maxcol = !D.N_COLORS
r_curr = bindgen(maxcol)
g_curr = r_curr
b_curr = r_curr

;   wht,grn,gry,wht,tur,blu,ylw,pur,blk,red
r_curr = [255, 0,211,255, 0, 0,255, 55, 0,255]
g_curr = [255,100,211,255,200, 0,255, 0, 0, 0]
b_curr = [255, 0,211,255,230,255, 0, 55, 0, 0]

*****
; to graph on the screen, must have the DISPLAY env set
*****
;TVLCT, r_curr, g_curr, b_curr

;integers
nColors = 0
nHeader = 0
nRecords = 0
nmatch = 0
fcstPer = 0

;floats
sng = 0.0
level1 = 0.0
minrange = 0.0
maxrange = 0.0

;strings
filename = "
str = "
strFormat = "
strHeader = "
strLegend = "
param = "
obType = "
geomName = "
```

```

modelName = "
name = "
yname = "

;structures
datLine = {dat, v_dtg:',', nobis:0, parm:',', units:',', $
            geom:',', typlvl:',', lvl_1:0.0, tau:0, $
            typstat:',', stat_val:0.0, v_src:',', obs_type:' '}

strFmt = '(a10, 2x, i5, a20, a15, a30, a15, f8.2, i5, a15, f8.2, a10, a25)'

;*****
; get the env vars and data filename
;*****
param = GETENV('PARM_NAME')
level1 = GETENV('LVL_1')
fcstPer = GETENV('FCSTPER')
obType = GETENV('OBSTYPE')
geomName = GETENV('GEOM_NAME')
modelName = GETENV('MODEL')
filename = GETENV('FILENAME')

; determine the number of records in the data file
data=READ_ASCII(fileName, count=nRecords)
print, "record count = ", nRecords

;array declarations
fields = replicate(datLine, nRecords)
bias = replicate(datLine, nRecords)
std = replicate(datLine, nRecords)
rms = replicate(datLine, nRecords)
std1 = FLTARR(nRecords)
std2 = FLTARR(nRecords)
strDTG = STRARR(nRecords)
lonDTG = FLTARR(nRecords, /NOZERO)
numObs = INTARR(nRecords)

openr, 10, filename

for n=0, nRecords-1 do begin
    READF, 10, datLine, FORMAT=strFmt
    fields[n] = datLine
endfor

close, 10

fields = fields(SORT(fields[*].v_dtg))

;*****
; get the sub arrays
;*****
bias = fields[WHERE((STRTRIM(fields[*].parm) EQ param) and $

```

```

(fields[*].lvl_1 EQ level1) and $
(STRTRIM(fields[*].geom) EQ geomName) and $
(fields[*].tau EQ fcstPer) and $
(STRTRIM(fields[*].obs_type) EQ obType) and $
(STRTRIM(fields[*].typstat) EQ 'bias', nmatch)]
start_date = JULDAY(STRMID(bias[0].v_dtg,4,2), $
STRMID(bias[0].v_dtg,6,2), $
STRMID(bias[0].v_dtg,0,4))
end_date = JULDAY(STRMID(bias[nmatch-1].v_dtg,4,2), $
STRMID(bias[nmatch-1].v_dtg,6,2), $
STRMID(bias[nmatch-1].v_dtg,0,4))
start_time = FIX(STRMID(bias[0].v_dtg,8,2)) / 24.
end_time = (end_date - start_date) $
+ (FLOAT(STRMID(bias[nmatch-1].v_dtg,8,2))) / 24.

for n=0, nmatch-1 do begin
end_date1 = JULDAY(STRMID(bias[n].v_dtg,4,2), $
STRMID(bias[n].v_dtg,6,2), $
STRMID(bias[n].v_dtg,0,4))
end_time1 = (end_date1 - start_date) $
+ (FLOAT(STRMID(bias[n].v_dtg,8,2))) / 24.
lonDTG[n] = (end_time1 - start_time) + start_time
numObs[n] = bias[n].nobs
endfor

std = fields[WHERE((STRTRIM(fields[*].parm) EQ param) and $
(fields[*].lvl_1 EQ level1) and $
(STRTRIM(fields[*].geom) EQ geomName) and $
(fields[*].tau EQ fcstPer) and $
(STRTRIM(fields[*].obs_type) EQ obType) and $
(STRTRIM(fields[*].typstat) EQ 'std', nmatch))]
rms = fields[WHERE((STRTRIM(fields[*].parm) EQ param) and $
(fields[*].lvl_1 EQ level1) and $
(STRTRIM(fields[*].geom) EQ geomName) and $
(fields[*].tau EQ fcstPer) and $
(STRTRIM(fields[*].obs_type) EQ obType) and $
(STRTRIM(fields[*].typstat) EQ 'rms', nmatch))]

for n=0, nmatch-1 do begin
std1[n] = bias[n].stat_val + std[n].stat_val
std2[n] = bias[n].stat_val - std[n].stat_val
endfor

dummy = LABEL_DATE(DATE_FORMAT = '%HZ %D%M %Z', offset=start_date)
i = nmatch - 1

;*****
;plot the data
;*****
;title string
;*****

```

```

if STRTRIM(bias[0].typ1vl) EQ 'isbr_1vl' then $
  name = modelName + ', ' + geomName + ', ' + param + '!C' $
  + STRTRIM(STRING(bias[0].1vl_1)) $
  + 'mb, ' + STRTRIM(STRING(fcstPer)) + ' hrs, ' + obType $
else $
  name = modelName + ', ' + geomName + ', ' + param + '!C' $
  + STRTRIM(STRING(bias[0].1vl_1)) $
  + 'm, ' + STRTRIM(STRING(fcstPer)) + ' hrs, ' + obType

yname = bias[i].units ; y-axis title string
minrange = MIN(bias[0:i].stat_val)
if (MIN(std2[0:i]) LT minrange) then $
  minrange = MIN(std2[0:i])
if (MIN(rms[0:i].stat_val) LT minrange) then $
  minrange = MIN(rms[0:i].stat_val)
maxrange = MAX(bias[0:i].stat_val)
if (MAX(std1[0:i]) GT maxrange) then $
  maxrange = MAX(std1[0:i])
if (MAX(rms[0:i].stat_val) GT maxrange) then $
  maxrange = MAX(rms[0:i].stat_val)

.*****
;
; for debugging
.*****
;
;print, "std"
;print, std[0:i].stat_val
;print, "rms"
;print, rms[0:i].stat_val
;print, "graph range lies between ", minrange, " and ", maxrange
;print, "lonDtg"
;print, lonDTG[0:i]
;print, "bias"
;print, bias[0:i]
;print, "std1"
;print, std1[0:i]
;print, "std2"
;print, std2[0:i]

!X.MINOR = -1 ; suppress minor tick marks
!Y.MARGIN(1) = 3 ; top margin

.*****
;
; to create a post script file
.*****
;
;set_plot, 'PS'
;psfile = filename + '.ps'
;device, /color, filename=psfile

.*****
;
; to create a gif file
.*****
;
;set_plot, 'Z'

```

```

psfile = filename + '.gif'

PLOT, lonDTG[0:i], bias[0:i].stat_val, $
  YRANGE = [minrange, maxrange], $
  TITLE = name, PSYM = -2, SYMSIZE = 1., $
  XTITLE = 'Forecast Date', $
  YTITLE = yname, $
  XGRIDSTYLE = 1, YGRIDSTYLE = 1, $
  XTICKLEN = 1.0, YTICKLEN = 1.0, $
  XTICKFORMAT = "label_date", $
  XCHARSIZE = 0.7, $
  MAX_VALUE = 30, $
  COLOR = 5, XSTYLE = 2, /DEVICE, /NODATA

for n=1,i do begin
  POLYFILL, [lonDTG(n-1), lonDTG[n-1:n], lonDTG(n)], $
    [bias[n-1].stat_val, std1[n-1:n], bias[n].stat_val], COLOR=2
endfor

for n=1,i do begin
  POLYFILL, [lonDTG(n-1), lonDTG[n-1:n], lonDTG(n)], $
    [bias[n-1].stat_val, std2[n-1:n], bias[n].stat_val], COLOR=2
endfor

; overplot the bias
OPLOT, lonDTG[0:i], bias[0:i].stat_val, PSYM = -2, SYMSIZE = 1., $
  LINE = 0, COLOR = 5, MAX_VALUE = 30

; overplot the std1
OPLOT, lonDTG[0:i], std1[0:i], PSYM = -6, SYMSIZE = 1., $
  LINE = 0, COLOR = 9, MAX_VALUE = 30

; overplot the std2
OPLOT, lonDTG[0:i], std2[0:i], PSYM = -6, SYMSIZE = 1., $
  LINE = 0, COLOR = 9, MAX_VALUE = 30

; overplot the rms
OPLOT, lonDTG[0:i], rms[0:i].stat_val, PSYM = -4, SYMSIZE = 1., $
  LINE = 0, COLOR = 7, MAX_VALUE = 30

; add the legends
XYOUTS, 0.8, 0.16, '15* - bias', color = 5, /NORMAL ; legend
XYOUTS, 0.8, 0.13, '1MB - std', color = 9, /NORMAL ; legend
XYOUTS, 0.8, 0.1, '1MV - rms', color = 7, /NORMAL ; legend

image = TVRD()
WRITE_GIF, psfile, image

;DEVICE, /close_file
SET_PLOT, 'X'

END

```

D. USER INTERFACE

1. Index.html

```
<html>
<!--
Author: Susie Pace
Date: 25 March 1998
File URL: model_reports/mverif/index.html
-->

<head><title>FNMOC Model Statistics Display </title></head>
<body bgcolor="#191970" TEXT="#F5F5DC" LINK="#00FF7F"
VLINK="#CCCC66" ALINK="#FF0000">
<FONT SIZE=+1>

<h1><center>Pick a model to see the statistics.</center></h1>
<UL>
  <A HREF="nogaps.html"> NOGAPS</A></P>
  <A HREF="noraps_asia.html">NORAPS_ASIA</A></P>
  <A HREF="noraps_conus.html">NORAPS_CONUS</A></P>
  <A HREF="noraps_europe.html">NORAPS_EUROPE</A></P>
  <A HREF="noraps_ind_ocn.html">NORAPS_IND_OCN</A></P>
  <A HREF="coamps_europe.html">COAMPS_EUROPE</A></P>
  <A HREF="coamps_swasia.html">COAMPS_SOUTHWEST_ASIA</A></P>
  <A HREF="wam.html">WAM_GLOBAL</A></P>
</UL>

<HR>
<CENTER><FONT size="-1"><I> Send Comments Or Suggestions To
Susie Pace: <A HREF="mailto:pacek@fnmoc.navy.mil">
pacek@fnmoc.navy.mil</A>
<BR> Last Update Was On March 25, 1998 </I></FONT></CENTER>

</BODY>
</HTML>
```

2. Nogaps.html

```
<html>
<!--
Author: Susie Pace
Date: 25 March 1998
File URL: model_reports/mverif/nogaps.html
-->

<head><title>NOGAPS Verification Display </title></head>
<body bgcolor="#191970" TEXT="#00ff7f" LINK="#00FF7F" VLINK="#CCCC66"
```

```

ALINK="#FF0000">
<form method=GET action="http://devul/cgi-bin/space3.pl">

<h1><center>Make your selections to see the statistics.</center></h1>

<h3>Model:</h3>
<input type=radio name=model value="nogaps" checked>nogaps

<h3>Geometry:</h3>
<input type=radio name=geometry value="global_360x181" checked>global_360x181
<input type=radio name=geometry value="asia_nest1_appl">asia_nest1_appl
<input type=radio name=geometry value="conus_nest1_appl">conus_nest1_appl
<input type=radio name=geometry value="europe_nest1_appl">europe_nest1_appl
<input type=radio name=geometry value="europe_nest2_appl2">europe_nest2_appl2
<input type=radio name=geometry value="europe_nest3_appl3">europe_nest3_appl3
<input type=radio name=geometry value="ind_ocn_nest1_appl">ind_ocn_nest1_appl
<!--<input type=radio name=geometry value="southwest_asia_nest2_appl">southwest_asia_nest2_appl
<input type=radio name=geometry value="southwest_asia_nest3_appl">southwest_asia_nest3_appl
-->

<h3>Parameters:</h3>
<input type=radio name=parameter value="air_temp">air_temp
<input type=radio name=parameter value="geop_ht" checked>geop_ht
<input type=radio name=parameter value="pres">pres
<input type=radio name=parameter value="wnd_spd">wnd_spd

<h3>Taus:</h3>
<input type=radio name=tau value="0">0
<input type=radio name=tau value="12">12
<input type=radio name=tau value="24" checked>24
<input type=radio name=tau value="36">36
<input type=radio name=tau value="48">48
<input type=radio name=tau value="60">60
<input type=radio name=tau value="72">72
<input type=radio name=tau value="84">84
<input type=radio name=tau value="96">96
<input type=radio name=tau value="108">108
<input type=radio name=tau value="120">120
<input type=radio name=tau value="132">132
<input type=radio name=tau value="144">144

<h3>Levels:</h3>
<input type=radio name=level value="0">0
<input type=radio name=level value="2">2
<input type=radio name=level value="19.5">19.5
<input type=radio name=level value="1000">1000
<input type=radio name=level value="925">925
<input type=radio name=level value="850">850
<input type=radio name=level value="700">700
<input type=radio name=level value="500" checked>500
<input type=radio name=level value="400">400
<input type=radio name=level value="300">300

```

```

<input type=radio name=level value="250">250
<input type=radio name=level value="200">200
<input type=radio name=level value="150">150
<input type=radio name=level value="100">100

<!--<h3>Statistics:</H3>
bias:      <input type=checkbox name="bias">
stdev:     <input type=checkbox name="stdev">
rms:       <input type=checkbox name="rms">
-->

<h3>Obs types:</h3>
<input type=radio name=obstype value="raob_qc" checked>raob_qc
<input type=radio name=obstype value="sfc_lnd">sfc_lnd
<input type=radio name=obstype value="sfc_ship_met_qc">sfc_ship_met_qc

<h3>Graph Type:</h3>
<!--<input type=radio name=graph value="scatter plot">scatter_plot-->
<input type=radio name=graph value="time series" checked>time_series

<h3>Period: Enter the beginning and ending DTG: e.g., 1998031812</h3>
<input type=text name=beginning maxlength=10 value="1998032512">
<input type=text name=ending maxlength=10 value="1998040500">

<input type=submit>
<input type=reset value="Cancel">

</form>
</body>
</html>

```

3. Noraps_asia.html

```

<html>
<!--
Author:  Susie Pace
Date:    07 November, 1997
File URL: model_reports/mverif/noraps_asia.html
-->

<head><title>NORAPS_ASIA Verification Display </title></head>
<body bgcolor="#191970" TEXT="#00FF7F" LINK="#00FF7F" VLINK="#CCCC66"
ALINK="#FF0000">
<form method=GET action="http://devu1/cgi-bin/space3.pl">

<h1><center>Make your selections to see the statistics.</center></h1>
<h3>Model:</h3>
<input type=radio name=model value="noraps_asia" checked>noraps_asia
<h3>Geometry:</h3>
<input type=radio name=geometry value="asia_nest1_appl" checked> asia_nest1_appl

```

```

<h3>Parameters:</h3>
<input type=radio name=parameter value="air_temp"> air_temp
<input type=radio name=parameter value="geop_ht" checked> geop_ht
<input type=radio name=parameter value="pres"> pres
<input type=radio name=parameter value="wnd_spd"> wnd_spd

<h3>Taus:</h3>
<input type=radio name=tau value="0"> 0
<input type=radio name=tau value="12"> 12
<input type=radio name=tau value="24" checked> 24
<input type=radio name=tau value="36"> 36
<input type=radio name=tau value="48"> 48

<h3>Levels: </h3>
<input type=radio name=level value="0">0
<input type=radio name=level value="2">2
<input type=radio name=level value="19.5">19.5
<input type=radio name=level value="1000">1000
<input type=radio name=level value="925">925
<input type=radio name=level value="850">850
<input type=radio name=level value="700">700
<input type=radio name=level value="500" checked>500
<input type=radio name=level value="400">400
<input type=radio name=level value="300">300
<input type=radio name=level value="250">250
<input type=radio name=level value="200">200
<input type=radio name=level value="150">150
<input type=radio name=level value="100">100

<!--<h3>Statistics:</H3>
bias:      <input type=checkbox name="bias">
stdev:     <input type=checkbox name="stdev">
rms:       <input type=checkbox name="rms">
-->

<h3>Obs types:</h3>
<input type=radio name=obstype value="raob_qc" checked>raob_qc
<input type=radio name=obstype value="sfc_lnd">sfc_lnd
<input type=radio name=obstype value="sfc_ship_met_qc">sfc_ship_met_qc

<h3>Graph Type:</h3>
<!--<input type=radio name=graph value="scatter plot">scatter_plot-->
<input type=radio name=graph value="time series" checked>time_series

<h3>Period: Enter the beginning and ending DTG: e.g., 1998031812</h3>
<input type=text name=beginning maxlength=10 value="1998031812">
<input type=text name=ending maxlength=10 value="1998033012">

<input type=submit>
<input type=reset value="Cancel">

</form>

```

```
</body>
</html>
```

4. Noraps_conus.html

```
<html>
<!--
Author: Susie Pace
Date: 25 March 1998
File URL: model_reports/mverif/noraps_conus.html
-->

<head><title>NORAPS_CONUS Verification Display </title></head>

<body bgcolor="#191970" TEXT="#00FF7F" LINK="#00FF7F" VLINK="#CCCC66"
ALINK="#FF0000">
<form method=GET action="http://devul/cgi-bin/space3.pl">

<h1><center>Make your selections to see the statistics.</center></h1>
<h3>Model: </h3>
<input type=radio name=model value="noraps_conus" checked> noraps_conus

<h3>Geometry:</h3>
<input type=radio name=geometry value="conus_nest1_appl" checked> conus_nest1_appl

<h3>Parameters:</h3>
<input type=radio name=parameter value="air_temp"> air_temp
<input type=radio name=parameter value="geop_ht" checked> geop_ht
<input type=radio name=parameter value="pres"> pres
<input type=radio name=parameter value="wnd_spd"> wnd_spd

<h3>Taus:</h3>
<input type=radio name=tau value="0"> 0
<input type=radio name=tau value="12"> 12
<input type=radio name=tau value="24" checked> 24
<input type=radio name=tau value="36"> 36
<input type=radio name=tau value="48"> 48

<h3>Levels: </h3>
<input type=radio name=level value="0">0
<input type=radio name=level value="2">2
<input type=radio name=level value="19.5">19.5
<input type=radio name=level value="1000">1000
<input type=radio name=level value="925">925
<input type=radio name=level value="850">850
<input type=radio name=level value="700">700
<input type=radio name=level value="500" checked>500
<input type=radio name=level value="400">400
<input type=radio name=level value="300">300
<input type=radio name=level value="250">250
<input type=radio name=level value="200">200
```

```

<input type=radio name=level value="150">150
<input type=radio name=level value="100">100

<!--<h3>Statistics:</H3>
bias:      <input type=checkbox name="bias">
stdev:     <input type=checkbox name="stdev">
rms:       <input type=checkbox name="rms">
-->

<h3>Obs types:</h3>
<input type=radio name=obstype value="raob_qc" checked>raob_qc
<input type=radio name=obstype value="sfc_lnd">sfc_lnd
<input type=radio name=obstype value="sfc_ship_met_qc">sfc_ship_met_qc

<h3>Graph Type:</h3>
<!--<input type=radio name=graph value="scatter plot">scatter_plot-->
<input type=radio name=graph value="time series" checked>time_series

<h3>Period: Enter the beginning and ending DTG: e.g., 1998031812</h3>
<input type=text name=beginning maxlength=10 value="1998031812">
<input type=text name=ending maxlength=10 value="1998033012">

<input type=submit>
<input type=reset value="Cancel">

</form>
</body>
</html>

```

5. Noraps_europe.html

```

<html>
<!--
Author:  Susie Pace
Date:    25 March 1998
File URL: model_reports/mverif/noraps_europe.html
-->

<head><title>NORAPS_EUROPE Verification Display </title></head>
<body bgcolor="#191970" TEXT="#00FF7F" LINK="#00FF7F" VLINK="#CCCC66"
ALINK="#FF0000">
<form method=GET action="http://devul/cgi-bin/space3.pl">

<h1><center>Make your selections to see the statistics.</center></h1>

<h3>Model: </h3>
<input type=radio name=model value="noraps_europe" checked> noraps_europe

<h3>Geometry:</h3>
<input type=radio name=geometry value="europe_nest1_appl" checked>europe_nest1_appl

```

```

<h3>Parameters:</h3>
<input type=radio name=parameter value="air_temp"> air_temp
<input type=radio name=parameter value="geop_ht" checked> geop_ht
<input type=radio name=parameter value="pres"> pres
<input type=radio name=parameter value="wnd_spd"> wnd_spd

<h3>Taus:</h3>
<input type=radio name=tau value="0"> 0
<input type=radio name=tau value="12"> 12
<input type=radio name=tau value="24" checked> 24
<input type=radio name=tau value="36"> 36
<input type=radio name=tau value="48"> 48

<h3>Levels: </h3>
<input type=radio name=level value="0">0
<input type=radio name=level value="2">2
<input type=radio name=level value="19.5">19.5
<input type=radio name=level value="1000">1000
<input type=radio name=level value="925">925
<input type=radio name=level value="850">850
<input type=radio name=level value="700">700
<input type=radio name=level value="500" checked>500
<input type=radio name=level value="400">400
<input type=radio name=level value="300">300
<input type=radio name=level value="250">250
<input type=radio name=level value="200">200
<input type=radio name=level value="150">150
<input type=radio name=level value="100">100

<!--<h3>Statistics:</H3>
bias:      <input type=checkbox name="bias">
stdev:     <input type=checkbox name="stdev">
rms:       <input type=checkbox name="rms">
-->

<h3>Obs types:</h3>
<input type=radio name=obstype value="raob_qc" checked>raob_qc
<input type=radio name=obstype value="sfc_lnd">sfc_lnd
<input type=radio name=obstype value="sfc_ship_met_qc">sfc_ship_met_qc

<h3>Graph Type:</h3>
<!--<input type=radio name=graph value="scatter plot">scatter_plot-->
<input type=radio name=graph value="time series" checked>time_series

<h3>Period: Enter the beginning and ending DTG: e.g., 1998031812</h3>
<input type=text name=beginning maxlength=10 value="1998031812">
<input type=text name=ending maxlength=10 value="1998033012">

<input type=submit>
<input type=reset value="Cancel">

</form>

```

```
</body>
</html>
```

6. Noraps_ind_ocn.html

```
<html>
<!--
Author: Susie Pace
Date: 25 March 1998
File URL: model_reports/mverif/noraps_ind_ocn.html
-->

<head><title>NORAPS_IND_OCN Verification Display </title></head>
<body bgcolor="#191970" TEXT="#00FF7F" LINK="#00FF7F" VLINK="#CCCC66"
ALINK="#FF0000">
<form method=GET action="http://devul/cgi-bin/space3.pl">

<h1><center>Make your selections to see the statistics.</center></h1>

<h3>Model: </h3>
<input type=radio name=model value="noraps_ind_ocn" checked>noraps_ind_ocn

<h3>Geometry:</h3>
<input type=radio name=geometry value="ind_ocn_nest1_appl" checked>
ind_ocn_nest1_appl

<h3>Parameters:</h3>
<input type=radio name=parameter value="air_temp"> air_temp
<input type=radio name=parameter value="geop_ht" checked> geop_ht
<input type=radio name=parameter value="pres"> pres
<input type=radio name=parameter value="wnd_spd"> wnd_spd

<h3>Taus:</h3>
<input type=radio name=tau value="0"> 0
<input type=radio name=tau value="12"> 12
<input type=radio name=tau value="24" checked> 24
<input type=radio name=tau value="36"> 36
<input type=radio name=tau value="48"> 48

<h3>Levels: </h3>
<input type=radio name=level value="0">0
<input type=radio name=level value="2">2
<input type=radio name=level value="19.5">19.5
<input type=radio name=level value="1000">1000
<input type=radio name=level value="925">925
<input type=radio name=level value="850">850
<input type=radio name=level value="700">700
<input type=radio name=level value="500" checked>500
<input type=radio name=level value="400">400
<input type=radio name=level value="300">300
<input type=radio name=level value="250">250
```

```

<input type=radio name=level value="200">200
<input type=radio name=level value="150">150
<input type=radio name=level value="100">100

<!--<h3>Statistics:</H3>
bias:      <input type=checkbox name="bias">
stdev:     <input type=checkbox name="stdev">
rms:       <input type=checkbox name="rms">
-->

<h3>Obs types:</h3>
<input type=radio name=obstype value="raob_qc" checked>raob_qc
<input type=radio name=obstype value="sfc_lnd">sfc_lnd
<input type=radio name=obstype value="sfc_ship_met_qc">sfc_ship_met_qc

<h3>Graph Type:</h3>
<!--<input type=radio name=graph value="scatter plot">scatter_plot-->
<input type=radio name=graph value="time series" checked>time_series

<h3>Period: Enter the beginning and ending DTG: e.g., 1998031812</h3>
<input type=text name=beginning maxlength=10 value="1998031812">
<input type=text name=ending maxlength=10 value="1998033012">

<input type=submit>
<input type=reset value="Cancel">

</form>
</body>
</html>

```

7. Coamps_europe.html

```

<html>
<!--
Author:  Susie Pace
Date:    25 March 1998
File URL: model_reports/mverif/coamps_europe.html
-->

<head><title>COAMPS_EUROPE Verification Display </title></head>
<body bgcolor="#191970" TEXT="#00FF7F" LINK="#00FF7F" VLINK="#CCCC66"
ALINK="#FF0000">
<form method=GET action="http://devu1/cgi-bin/space3.pl">

<h1><center>Make your selections to see the statistics.</center></h1>

<h3>Model: </h3>
<input type=radio name=model value="coamps_europe" checked> coamps_europe

<h3>Geometry:</h3>
<input type=radio name=geometry value="europe_nest2_appl2" checked>

```

europe_nest2_appl2
<input type=radio name=geometry value="europe_nest3_appl3">europe_nest3_appl3

Parameters:</h3>

<input type=radio name=parameter value="air_temp"> air_temp
<input type=radio name=parameter value="geop_ht" checked> geop_ht
<input type=radio name=parameter value="pres"> pres
<input type=radio name=parameter value="wnd_spd"> wnd_spd

Taus:</h3>

<input type=radio name=tau value="0"> 0
<input type=radio name=tau value="12"> 12
<input type=radio name=tau value="24" checked> 24
<input type=radio name=tau value="36"> 36
<input type=radio name=tau value="48"> 48

Levels: </h3>

<input type=radio name=level value="0">0
<input type=radio name=level value="2">2
<input type=radio name=level value="19.5">19.5
<input type=radio name=level value="1000">1000
<input type=radio name=level value="925">925
<input type=radio name=level value="850">850
<input type=radio name=level value="700">700
<input type=radio name=level value="500" checked>500
<input type=radio name=level value="400">400
<input type=radio name=level value="300">300
<input type=radio name=level value="250">250
<input type=radio name=level value="200">200
<input type=radio name=level value="150">150
<input type=radio name=level value="100">100

Statistics:</H3>

bias: <input type=checkbox name="bias">
stdev: <input type=checkbox name="stdev">
rms: <input type=checkbox name="rms">
-->

Obs types:</h3>

<input type=radio name=obstype value="raob_qc" checked>raob_qc
<input type=radio name=obstype value="sfc_lnd">sfc_lnd
<input type=radio name=obstype value="sfc_ship_met_qc">sfc_ship_met_qc

Graph Type:</h3>

<!--<input type=radio name=graph value="scatter plot">scatter_plot-->
<input type=radio name=graph value="time series" checked>time_series

Period: Enter the beginning and ending DTG: e.g., 1998031812</h3>

<input type=text name=beginning maxlength=10 value="1998031812">
<input type=text name=ending maxlength=10 value="1998033012">

<input type=submit>

```
<input type=reset value="Cancel">
```

```
</form>
```

```
</body>
```

```
</html>
```

8. Coamps_southwest_asia.html

```
<html>
```

```
<!--
```

```
Author: Susie Pace
```

```
Date: 25 March 1998
```

```
File URL: model_reports/mverif/coamps_swasia.html
```

```
-->
```

```
<head><title>COAMPS_SOUTHWEST_ASIA Verification Display </title></head>
```

```
<body bgcolor="#191970" TEXT="#00FF7F" LINK="#00FF7F" VLINK="#CCCC66"  
ALINK="#FF0000">
```

```
<form method=GET action="http://devul/cgi-bin/space3.pl">
```

```
<h1><center>Make your selections to see the statistics.</center></h1>
```

```
<h3>Model: </h3>
```

```
<input type=radio name=model value="coamps_sw_asia" checked>  
coamps_southwest_asia
```

```
<h3>Geometry: </h3>
```

```
<input type=radio name=geometry value="southwest_asia_nest2_appl" checked>  
southwest_asia_nest2_appl  
<input type=radio name=geometry value="southwest_asia_nest3_appl">  
southwest_asia_nest3_appl
```

```
<h3>Parameters: </h3>
```

```
<input type=radio name=parameter value="air_temp"> air_temp  
<input type=radio name=parameter value="geop_ht" checked> geop_ht  
<input type=radio name=parameter value="pres"> pres  
<input type=radio name=parameter value="wnd_spd"> wnd_spd
```

```
<h3>Taus: </h3>
```

```
<input type=radio name=tau value="0"> 0  
<input type=radio name=tau value="12"> 12  
<input type=radio name=tau value="24" checked> 24  
<input type=radio name=tau value="36"> 36  
<input type=radio name=tau value="48"> 48
```

```
<h3>Levels: </h3>
```

```
<input type=radio name=level value="0">0  
<input type=radio name=level value="2">2  
<input type=radio name=level value="19.5">19.5  
<input type=radio name=level value="1000">1000  
<input type=radio name=level value="925">925
```

```

<input type=radio name=level value="850">850
<input type=radio name=level value="700">700
<input type=radio name=level value="500" checked>500
<input type=radio name=level value="400">400
<input type=radio name=level value="300">300
<input type=radio name=level value="250">250
<input type=radio name=level value="200">200
<input type=radio name=level value="150">150
<input type=radio name=level value="100">100

<!--<h3>Statistics:</H3>
bias:      <input type=checkbox name="bias">
stdev:     <input type=checkbox name="stdev">
rms:       <input type=checkbox name="rms">
-->

<h3>Obs types:</h3>
<input type=radio name=obstype value="raob_qc" checked>raob_qc
<input type=radio name=obstype value="sfc_lnd">sfc_lnd
<input type=radio name=obstype value="sfc_ship_met_qc">sfc_ship_met_qc

<h3>Graph Type:</h3>
<!--<input type=radio name=graph value="scatter plot">scatter_plot-->
<input type=radio name=graph value="time series" checked>time_series

<h3>Period: Enter the beginning and ending DTG: e.g., 1998031812</h3>
<input type=text name=beginning maxlength=10 value="1998031812">
<input type=text name=ending maxlength=10 value="1998033012">

<input type=submit>
<input type=reset value="Cancel">

</form>
</body>
</html>

```

9. Wam_Global.html

```

<html>
<!--
Author:  Susie Pace
Date:    25 March 1998
File URL: model_reports/mverif/wam.html
-->

<head><title>WAM_GLOBAL Verification Display </title></head>
<body bgcolor="#191970" TEXT="#00FF7F" LINK="#00FF7F" VLINK="#CCCC66"
ALINK="#FF0000">
<form method=GET action="http://devul/cgi-bin/space3.pl">

<h1><center>Make your selections to see the statistics.</center></h1>

```

```

<h3>Model: </h3>
<input type=radio name=model value="wam_global" checked> wam_global

<h3>Geometry:</h3>
<input type=radio name=geometry value="global_360x181" checked>
global_360x181

<h3>Parameters:</h3>
<input type=radio name=parameter value="sig_wav_ht" checked> sig_wav_ht
<input type=radio name=parameter value="peak_wav_per"> peak_wav_per

<h3>Taus:</h3>
<input type=radio name=tau value="0"> 0
<input type=radio name=tau value="12"> 12
<input type=radio name=tau value="24" checked> 24
<input type=radio name=tau value="36"> 36
<input type=radio name=tau value="48"> 48
<input type=radio name=tau value="60"> 60
<input type=radio name=tau value="72"> 72
<input type=radio name=tau value="84"> 84
<input type=radio name=tau value="96"> 96
<input type=radio name=tau value="108"> 108
<input type=radio name=tau value="120"> 120

<h3>Levels: </h3>
<input type=radio name=level value="0" checked>0

<!--<h3>Statistics:</H3>
bias:    <input type=checkbox name="bias">
stdev:   <input type=checkbox name="stdev">
rms:     <input type=checkbox name="rms">
-->

<h3>Obs types:</h3>
<input type=radio name=obstype value="sfc_ship" checked>sfc_ship
<!--<input type=radio name=obstype value="alty">alty-->

<h3>Graph Type:</h3>
<!--<input type=radio name=graph value="scatter plot">scatter_plot-->
<input type=radio name=graph value="time series" checked>time_series

<h3>Period: Enter the beginning and ending DTG: e.g., 1998031812</h3>
<input type=text name=beginning maxlength=10 value="1998031812">
<input type=text name=ending maxlength=10 value="1998033012">

<input type=submit>
<input type=reset value="Cancel">

</form>
</body>
</html>

```

10. Procform.pl

```
#!/usr/local/bin/perl
# space3.pl - Try to batch in runjob to retrieve the data
# and run the IDL programs to create the graphs

#Use the form library
require "space2.pl";

##### STEP 1: Get and decode the input from the form #####

#get the data from the form
&ReadParse(*input);

##### STEP 2: Process the information from the form #####

$procid = $$;
`export procid`;

#split $query_string into name=value pairs
local(*FormData) = @_ if @_; #make an alias for the arg
$query_string=$ENV{'QUERY_STRING'};

foreach $name_value (split('&', $query_string)) {

    #translate any plus signs in the pair string into spaces:
    $name_value =~ tr/+//;

    #split the name=value pair into a separate name and value:
    ($name, $value) = split('=', $name_value);

    #translate escaped hex numbers back to 8-bit char:
    $name =~ s/%(..)/pack("C", hex($1))/eg;
    $value =~ s/%(..)/pack("C", hex($1))/eg;

    #convert the names and values into named, assigned var;
    if (defined($FormData{$name})) {
        $FormData{$name} .= "\0$value";
    } else {
        $FormData{$name} = $value;
    }

    if ($name eq 'model') {
        $model = $value;
        `export model`;
        #exit 0 if ($model eq "")

    } elsif ($name eq 'geometry') {
        $geometry = $value;
        `export geometry`;
        #exit 0 if ($geometry eq "")
    }
}
```

```

} elif ($name eq 'parameter') {
$parameter = $value;
`export parameter`;
#exit 0 if ($parameter eq "")

} elif ($name eq 'tau') {
$tau = $value;
`export tau`;
#exit 0 if ($tau eq "")

} elif ($name eq 'level') {
$level = $value;
`export level`;
#exit 0 if ($level eq "")

} elif ($name eq 'stats') {
$stats = $value;
#`export stats`;

} elif ($name eq 'obstype') {
$obstype = $value;
`export obstype`;
#exit 0 if ($obstype eq "")

} elif ($name eq 'graph') {
$graph = $value;
`export graph`;
#exit 0 if ($graph eq "")

} elif ($name eq 'beginning') {
$beginning = $value;
`export beginning`;
#exit 0 if ($beginning eq "")

} elif ($name eq 'ending') {
$ending = $value;
`export ending`;
#exit 0 if ($ending eq "")

} else {
    print "no matching env var";
}

}

#process the request by runjob
`/a/ops/bin/runjob -h div60-3 -u pacek -t sun -d /home/pubs43/tmp -j get_data.ksh -e "pid=$procid
MODEL=$model BDTG=$beginning EDTG=$ending PARM_NAME=$parameter LVL_1=$level
FCSTPER=$tau OBSTYPE=$obstype GEOM_NAME=$geometry" > $procid.out 2>&1`;

```

STEP 3: Reply by outputting a new document

```

#$Image = "http://152.80.13.201/~pacek/gif/$procid.gif";
$image = "/home/pubs43/tmp/$procid.gif";

#!!!! This portion worked for WAM !!!!#
#
# Output the MIME-type header, followed by two newlines:
print "Content-type: image/gif\n\n";
#
# loop until the gif file is transferred to the web server
# when the gif file arrives, display on the web browser
#
while (! (-e $Image)) {
    sleep 5;
}

open (IMAGE, $Image);
print <IMAGE>;
close (IMAGE);

`rm $Image`;

#print "<HTML><HEAD><TITLE> Model Verification Reply </TITLE></HEAD>\n";
#print "</BODY></HTML>\n";
#
#!!!! Down to here !!!!#

#Trying to handle the timed out error

#if (-e $Image) {
# print "Content-type: image/gif\n\n";
# open (IMAGE, $Image);
# print <IMAGE>;
# close (IMAGE);
#this causes the runjob to be batched in multiple time and PID change,
#end up in an infinite loop
#} else {
# print "refresh: 10; \n";
# print "Content-type: text/html\n\n";
# print "<HTML><HEAD><TITLE>Model Statistics Display</TITLE></HEAD>\n";
# print "<BODY BGCOLOR=NAVY TEXT=WHITE>";
# print "<H1><CENTER>Please be patient, the image is being created</CENTER></H1>\n";
# print "Processing is done when a graph is displayed\n";
#}

##### END OF space3.pl #####

```

11. Get_data.ksh

```

#!/bin/sh
/bin/ksh <<'EOT'
./a/ops/isis/db_init/isis_init_ofs.ksh

```

```

#set -x
cd /d/tmp
export DB_DIR=/d/model-stats/
cd $DB_DIR

export FILENAME=$pid
export TMPFN=$pid"tmp"
SLASH=_

if [[ $MODEL = nogaps ]]
then
    MODEL="$MODEL$SLASH$GEOM_NAME"
fi

print $MODEL $GEOM_NAME $FCSTPER $PARM_NAME $OBSTYPE $LVL_1 $BDTG $EDTG

empcmd $DB_DIR/stat_db 'select bypass_lock verif_date,sample_size,param_name,
unit_name,geom_name,lvl_type,level_1 convert to decimal(8,2),
tau,stat_type,stat_value convert to decimal(8,2),verif_source,
obs_type from $MODEL where verif_date range $BDTG to $EDTG and
parm_name="$PARM_NAME" and geom_name="$GEOM_NAME" and tau="$FCSTPER"
and obs_type="$OBSTYPE" and level_1="$LVL_1" into "$TMPFN";'

awk '/^[0-9]/\
{printf ("%12s%-5d%-20s%-15s%-30s%-15s%-8.2f%-5d%-15s%-8.2f%-10s%-25s\n", \
$1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12)}' $TMPFN > $FILENAME

#print "sourcing the IDL setup script"
./idl/idl_5/bin/idl_setup.ksh

#print $PARM_NAME $LVL_1 $FCSTPER $OBSTYPE $GEOM_NAME
idl -rt=plot_data

if [[ -f $FILENAME.gif ]]
then
    ftpbatch -h devu1 -s "cd /home/pubs43/tmp/; put $FILENAME.gif"
fi

rm $pid*

exit 0
EOT
exit 0

```

APPENDIX C

A. STATTEST.F90

```
program stattest
C
C.....START PROLOGUE.....
C
C SCCS IDENTIFICATION: @(#)stattest.f90      1.1 04/24/98
C
C CONFIGURATION IDENTIFICATION: NONE
C
C MODULE NAME: stattest
C
C DESCRIPTION: Program to test the stat lib
C
C COPYRIGHT:      (c) 1998 FLENUMMETOCCEN
C      U.S. GOVERNMENT DOMAIN
C      ALL RIGHTS RESERVED
C
C CONTRACT NUMBER AND TITLE: N/A
C
C REFERENCES: NONE
C
C CLASSIFICATION: Unclassified
C
C RESTRICTIONS: NONE
C
C COMPUTER/OPERATING SYSTEM
C      DEPENDENCIES: Cray UNICOS
C
C LIBRARIES OF RESIDENCE: /a/ops/bin
C
C USAGE:
C  stattest
C
C PARAMETERS:
C   Name      Type      Usage      Description
C  -----
C
C PARAMETERS:
C   Name      Type      Usage      Description
C  -----
C
C COMMON BLOCKS: N/A
C
C FILES: N/A
C
C DATA BASES: N/A
```

```

C
C COMPILER DEPENDENCIES: f90
C
C COMPILE OPTIONS: -f fixed -c
C
C MAKEFILE: Located at /a/ops/app/mverif/test/maketest
C      UNICOS make
C
C RECORD OF CHANGES:
C
C <<CHANGE NOTICE>> Version 1.1 (29 Apr 1998) -- Kyongsuk Pace
C      Initial submission
C
C.....END PROLOGUE.....
C

```

```

      implicit none

```

```

      integer :: size
      parameter(size=10)

```

```

      real  :: stat
      real  :: arr1(size)
      real  :: arr2(size)

```

```

      character(4) :: geom

```

```

      geom(1:4) = 'NONE'

```

```

      arr2 = (/288.6, 304.8, 301.4, 293.2, 293.2,
2      297.6, 295.8, 291.0, 285.2, 287.8/)
      arr1 = (/289.29, 302.26, 302.14, 294.92, 294.92,
2      296.25, 295.37, 292.32, 285.15, 288.69/)

```

```

      CALL COMPUTE_BIAS(arr1,arr2,size,geom,stat)
      write (0, '("bias = ", f8.2)') stat

```

```

      CALL COMPUTE_STD(arr1,arr2,size,geom,stat)
      write (0, '("std = ", f8.2)') stat

```

```

      CALL COMPUTE_RMS(arr1,arr2,size,geom,stat)
      write (0, '("rms = ", f8.2)') stat

```

```

      stop 'Normal End'

```

```

      end

```

B. STATISTICS BY STATISTICS LIBRARY SUBROUTINES

```

Script started on Wed Feb 11 17:21:26 1998
[/dev/tty017]
s{j91}/home/pacek/mverif/test>tstattest

```

```

bias = 0.27
std = 1.31
rms = 1.34
STOP Normal End
STOP executed at line 28 in Fortran routine 'STATTEST'
CPU: 0.010s, Wallclock: 0.011s, 10.6% of 8-CPU Machine
Memory HWM: 1595052, Stack HWM: 803866, Stack segment expansions: 0
{j91}/home/pacek/mverif/test
Script finished on Wed Feb 11 17:21:34 1998

```

C. STATISTICS BY MICROSOFT EXCEL

array 1	array 2	diff	diff	sqr
288.6	289.29	-0.69	0.4761	
304.8	302.26	2.54	6.4516	
301.4	302.14	-0.74	0.5476	
293.2	294.92	-1.72	2.9584	
293.2	294.92	-1.72	2.9584	
297.6	296.25	1.35	1.8225	
295.8	295.37	0.43	0.1849	
291	292.32	-1.32	1.7424	
285.2	285.15	0.05	0.0025	
287.8	288.69	-0.89	0.7921	
sum	-2.71	17.9365		
bias	-0.271	sum/count		
count	10			
stdev		1.3115	sqrt(sum of diff sqr/count - bias*bias)	
rms		1.3392	sqrt(sum of diff sqr/count)	

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center 8725 John J. Kingman Rd., STE 0944 Ft. Belvoir, VA 22060-6218	2
2. Dudley Knox Library Naval Postgraduate School 411 Dyer Rd. Monterey, CA 93943-5101	2
3. Dr. Dan Boger, Chairman, Code CS/Bo Department of Computer Science Naval Postgraduate School Monterey, CA 93943-5100	1
4. Chief of Naval Research 800 N. Quincy Street Arlington, VA 22217	1
5. Dr. Tim Shimeall, Code CS/Sm Department of Computer Science Naval Postgraduate School Monterey, CA 93943-5100	1
6. Dr. Mary Alice Rennick METOC Models Division, Code 410 Fleet Numerical Meterology and Oceanography Center 7 Grace Hopper Ave, Stop 1 Monterey, CA 93943-5501	1
7. Ms. Kyongsuk (Susie) Pace METOC Models Division, Code 410 Fleet Numerical Meterology and Oceanography Center 7 Grace Hopper Ave, Stop 1 Monterey, CA 93943-5501	3