NOAA Technical Report NOS CS 22

# COASTAL OCEAN MODELING FRAMEWORK: COMF

Silver Spring, Maryland
February 2006

noaa National Oceanic and Atmospheric Administration

U.S. DEPARTMENT OF COMMERCE
National Ocean Service
Coast Survey Development Laboratory

**Office of Coast Survey**
**National Ocean Service**
**National Oceanic and Atmospheric Administration**
**U.S. Department of Commerce**

**The Office of Coast Survey (CS) is the Nation's only official chartmaker. As the oldest United States scientific organization, dating from 1807, this office has a long history. Today it promotes safe navigation by managing the National Oceanic and Atmospheric Administration's (NOAA) nautical chart and oceanographic data collection and information programs.**

**There are four components of CS:**

> **The Coast Survey Development Laboratory develops new and efficient techniques to accomplish Coast Survey missions and to produce new and improved products and services for the maritime community and other coastal users.**
>
> **The Marine Chart Division collects marine navigational data to construct and maintain nautical charts, Coast Pilots, and related marine products for the United States.**
>
> **The Hydrographic Surveys Division directs programs for ship and shore-based hydrographic survey units and conducts general hydrographic survey operations.**
>
> **The Navigation Services Division is the focal point for Coast Survey customer service activities, concentrating predominantly on charting issues, fast-response hydrographic surveys and Coast Pilot updates.**

NOAA Technical Report NOS CS 22

# COASTAL OCEAN MODELING FRAMEWORK: COMF

Thomas F. Gross
Hong Lin
Office of Coast Survey

Zachary Bronder
Mark Vincent
Center for Operational Oceanographic Products and Services

February 2006

noaa National Oceanic and Atmospheric Administration

U.S. DEPARTMENT OF COMMERCE
Carlos Gutierrez, Secretary

National Oceanic and Atmospheric Administration
Conrad C. Lautenbacher, Jr., VADM USN (Ret.), Under Secretary

National Ocean Service
John H. Dunnigan, Assistant Administrator

Office of Coast Survey
Captain Roger L. Parsons, NOAA

Coast Survey Development Laboratory
Mary Erickson

NOTICE

Mention of a commercial company or product does not constitute an endorsement by NOAA. Use for publicity or advertising purposes of information from this publication concerning proprietary products or the tests of such products is not authorized.

## TABLE OF CONTENTS

LIST OF FIGURES ..... v

LIST OF TABLES ..... v

ABSTRACT ..... vii

1. INTRODUCTION ..... 1
   1.1 What is COMF? ..... 1
   1.2 Overview of COMF Modules for Operational Forecast Systems ..... 2
   1.3 The Future ..... 3

2. COMPONENTS OF THE COMF ..... 5
   2.1 Data Bank ..... 5
   2.2 Standardized Database Readers ..... 5
   2.3 CSDL Modelers Library ..... 6
   2.4 Standardized Outputs ..... 6
      2.4.1 Graphics ..... 6
      2.4.2 Web Pages ..... 6
      2.4.3 CORMS ..... 7
      2.4.4 Model Skill Assessment Tool ..... 7

3. DATA ACCESS METHODS ..... 9
   3.1 Readers and Variables ..... 9
      3.1.1 Readers and Variables ..... 9
      3.1.2 Time Series Output files: TS1, TS2, and TS3 ..... 11
      3.1.3 Ramp Filling for Water Level ..... 11
      3.1.4 Forecast Wind Field Access ..... 13
      3.1.5 Forecast Station Wind Reader: NAMSTATION ..... 13
   3.2 CORMS Flags ..... 14
   3.3 Data Banks ..... 16
      3.3.1 NWLON ..... 16
      3.3.2 NWLONweb ..... 16
      3.3.3 NDBC ..... 16
      3.3.4 USGS ..... 16
      3.3.5 ETSS ..... 17
      3.3.6 NAM ..... 17
      3.3.7 Other NCEP models ..... 17

4. SYSTEM DESCRIPTION ..... 19
   4.1 Directory Structure ..... 19
   4.2 Module 0: Environment Variables ..... 21
   4.3 Module 1: Computer System Tests ..... 22
   4.4 Module 2: Model Timing ..... 22
   4.5 Module 3: Data Access Tools ..... 23

4.6 Module 4: Reformat Data ..... 23
4.7 Module 5: Run the Hydrodynamic Model ..... 24
4.8 Module 6: Archive the Results ..... 24
4.9 Module 7: Make the Graphics ..... 25
4.10 Module 8: Make the CORMS Flags ..... 25
4.11 Module 9: Purge old files ..... 26
4.12 Web Pages ..... 26
4.13 Skill Assessment ..... 26

5. OUTPUT FILE STANDARDS ..... 27
5.1 NetCDF standard output format ..... 27
5.2 Log files ..... 27

6. SCRIPTS AND PROGRAM DESCRIPTIONS ..... 29
6.1 Scripts Library ..... 29
6.2 FORTRAN Library ..... 29

7. CVS AND TESTING ENVIRONMENT ..... 31

8. CONCLUSIONS ..... 35

ACKNOWLEGEMENTS ..... 35

REFERENCES ..... 37

APPENDIX A. BUILD A MODEL WITH COMF ..... 39

APPENDIX B. SCRIPTS LIBRARY ..... 51

APPENDIX C. FORTRAN LIBRARY ..... 109

APPENDIX D. SAMPLE MODEL MAIN SCRIPT (CBOFS) ..... 135

## LIST OF FIGURES

Figure 1 WLQCF.sh Missing Data Ramp Method. ..... 13
Figure 2 Sample of CORMS flags web page for an OFS. ..... 15

## LIST OF TABLES

Table 1 Access database. ..... 9
Table 2 Data grabber scripts description. ..... 10
Table 3 *QCF.sh calling parameters. ..... 10
Table 4 COMF-based system directories description. ..... 20
Table 5 Subdirectories of Operational Forecast System or the OQCS. ..... 21
Table 6 Input files of archive. ..... 24
Table 7 Output files of archive. ..... 24

**ABSTRACT**

The Coastal Ocean Modeling Framework (COMF) is an end-to-end set of standards and tools for NOAA National Ocean Service's (NOS) operational hydrodynamic forecast models. These models are created by Coast Survey Development Laboratory (CSDL) and run in the Center for Operational Oceanographic Products and Services (CO-OPS) operational environment. The usage of COMF by all NOAA/NOS operational models will allow a multiplicity of models to be maintained in an efficient and robust manner. The framework consists of standards and implementation of the standards for methods to read a variety of data sources to run a real-time modeling forecast system. The set of middleware provides a common look to all the data sources so that models can be easily developed, maintained and enhanced in the future. By standardizing operational models, great efficiency is achieved in building and testing. This should allow NOS to develop and implement forecast models, faster and of higher quality.

**Key words:** Operational nowcast/forecast systems, hydrodynamic forecast models, database reader, research to operation transitions, middleware, HYDRONetCDF, Oceanographic model, short-time forecast guidance

# 1. INTRODUCTION

NOAA and NOS have the mission and mandates to provide comprehensive coverage of predictions and information to support navigation and coastal needs[1 2 3]. To support this mission, NOS develops and maintains hydrodynamic model-based Operational Forecast Systems (OFS) for sea ports, estuaries, Great Lakes, and coastal water bodies, which together with NOAA's operational oceanographic capabilities form a national backbone of real-time data, tidal predictions, data management, and operational modeling. Critical factors that will enable NOS to successfully meet this requirement for OFS are: focused research resulting in validated enhancements; cost and time efficient development and production; and robust and reliable operations. The integrated solution to these collective needs is NOS's Coastal Ocean Modeling Framework (COMF).

## 1.1 What is COMF?

The Coastal Ocean Modeling Framework (COMF) is a set of standards and tools for developing and maintaining NOS's hydrodynamic model-based Operational Forecast Systems. The goal of COMF is to provide a comprehensive software infrastructure to increase ease of use, performance, portability, interoperability, and reuse in forecast models applied to models of estuaries, coast ocean and the Great Lakes and to provide a common interface to other NOAA (e.g. Earth System Modeling Framework – ESMF) and extramural partners and stakeholders. COMF provides a software framework for individual scientists, model production, and the critical operational environment. COMF is an absolute necessity for NOS to successfully support NOAA's mission goals and become a leader in estuarine and coastal modeling.

The net effect of COMF will be increased time-and-cost efficiency for forecast system development and production, combined with increased reliability for operations and maintenance. Best methods have, and will continue to be infused seamlessly into the standardized COMF components, enabling the community sharing of validated improvements and the minimizing of redundant parallel efforts.

All model forecast systems developed and produced for transfer to NOS operational status will be standardized within COMF. This will be accomplished by providing standard tools to perform as many of the modeling tasks as possible. An operational forecast system consists of a numerical hydrodynamic model code to calculate water levels, currents, water temperature and salinity. In addition, it includes code to access data sources and reformat the data for ingestion to the model code. Finally, output files from the model run are generated, disseminated and plotted for web pages. The data access task is always more difficult than the oceanographer anticipates and has historically resulted in excessive development times. It is also the weakest part of the operational model system as it is prone to errors, often breaks down due to changes in external data bases and requires continual maintenance in the operational environment. As the number of modeling systems increase, the development and maintenance cost of multiple versions of these systems becomes

[1] Section 883b of the Coast and Geodetic Survey Act of 1947 (33 U.S.C. 883a-i)
[2] Hydrographic Services Improvement Act of 1998 (33 U.S.C. 892 et seq.)
[3] Section 204(a)(2) of the High Performance Computing and Communication Act of 1991 (P.L. 102-994, 15 U.S.C. 55015528)

prohibitive. The COMF solves most of these problems by providing a complete suite of "middleware" between the databases and the models, and by enforcing standards for the operational models used by NOAA. In addition, standards are created to unify the output of the models so that only one set of products, graphics and web page programs are maintained.

The COMF is comprised of a collection of middleware that ensures uniform methods are used for all operational models to access data sources and produce outputs. The COMF components are composed of UNIX scripts (Bourne shell), FORTRAN programs, PERL scripts, and IDL for graphics. The new systems will consist of ten logically and simply defined modules which will be found in each model main script. Throughout this document an example of the Chesapeake Bay Operational Forecast System (CBOFS) main script, MAIN_CBOFS.sh (Appendix D) is used to demonstrate the usage of the modules and other scripting techniques.

## 1.2 Overview of COMF Modules for Operational Forecast Systems

The operational forecast systems are computer code which controls the timing, acquisition of data, running of the models, generation of site specific output, generation of graphics and dissemination of results via a web interface. In COMF, most of these capabilities are transparent to the model developer. The primary interface for each modeler is via a main shell script which runs automatically many times a day. The main shell run script (MAIN_**OFS.sh) consists of ten sections or modules. The sequential execution of these ten modules makes one full run cycle of the model.

Module 0: Set Environment variables for directories

Module 1: Computer system tests

Module 2: Create the start and stop times

Module 3: Get data
The series of programs pull the necessary external data off the various databases and make it available to the model.

Module 4: Reformat data
The series of programs to reformat the standard data into specific formats should be model independent. A set of standard formats (NetCDF) should be developed and used for all model input sources.

Module 5: Run the hydrodynamic model
The specific model code (Princeton Ocean Model (POM), Regional Ocean Modeling System (ROMS), Quoddy, Elcirc, etc.) is run. An alteration to the specific model code will be the usage of our standardized output methods to create the HYDRONetCDF files.

Module 6: Archive the data
The primary archive product will be the HYDRONetCDF output files. And others may be archived as well.

Module 7: Make the graphics
Standardized plotting programs will run on the standard HYDRONetCDF output files to create graphics which will be sent to standardized web pages for display by CO-OPS.

Module 8: Make the CORMS FLAGS
Standardized methods to monitor the flow of data through the model are used by the CO-OPS Continuous Operational Real-time Monitoring System (CORMS) group.

Module 9: Purge old files
Standardized script with an associated control file is used to routinely purge various file types. Old model archive and log files will be removed to conserve disk space.

The multiple modules give form and standardization to the operational forecast systems and simplify their creation. Of the module components of COMF the most significant and probably the most complicated is the unified data access methods, known as the Operational Quality Control System (OQCS). Unified data access is accomplished by providing a complete collection of data access tools which have been designed and tested to grab data from just a few sources. They provide the minimal forcing data set for our models of water level, wind, river discharge, salinity and temperature. These are obtained from the National Water Level Observation Network (NWLON) or Physical Oceanographic Real-Time System (PORTS) stations and a few other data sources, US Geological Survey (USGS) rivers, and National Data Buoy Center (NDBC) Coastal-Marine Automated Network (C-MAN) stations. NOAA forecast model guidance from North American Mesoscale meteorology model (NAM) and Extra Tropical Storm Surge (ETSS) models are also accessed with the same routines. Future models will only use these tools and all database management, calibration issues, quality control, CORMS flag generation and reformatting will be removed from the modeler's burden, allowing them to focus more efficiently on model application and validation.

### 1.3 The Future

COMF will be a dynamic framework that will infuse seamlessly validated enhancements and new techniques as standardized components. All enhancements will be required to pass rigorous testing and validation, and will be added to COMF using configuration management software. Formal version upgrades are targeted for release on an approximately annual schedule. There will be compromises and limitations placed upon the modelers and their development choices. But by careful design we can limit these and make the payoff in efficiency and quality well worth any initial inconveniences.

## 2. COMPONENTS OF THE COMF

COMF consists of more than just the code to make a model run. It is a collection of tools to aid in the whole process of developing an operational hydrodynamic model forecast system. It is an end-to-end system covering the model's data needs, running environment, auxiliary programming support, output file standardization, graphics, web pages, assessment and evaluation, model design and development.

### 2.1 Data Bank

The Data Bank is the collective repository of the real time data and the various operational forecast products which are necessary to run our models. There is no single computer and no single hard disk drive which is the data bank for COMF. Rather models access data from a variety of sources, some local, some from the Internet, some from archives and some from real-time data sources. The Operational Data Acquisition and Archiving System (ODAAS) is a large part of this, providing access to NWS forecast products (Kelley et al, 2001; Westington and Kelley 2003). The CO-OPS National Water Level Observation Program, NWLON, water level and PORTS database is another integral part (Evans, French and Bethem 1997; Bethem 1998; Nault 2000; Burton 2000). But some databases might not actually reside in CSDL or CO-OPS. For instance, direct reading of data from web sites, which provide sufficient flexibility, would be better than trying to recreate and maintain extensive, duplicative databases. In fact the old method of daily downloads via FTP to a local data base has been found to be less reliable than direct access to web sites, which are vigorously maintained as agencies' primary data distribution tools. Databases of this sort include the NDBC buoy and C-MAN stations, the USGS river stage and discharge database, and the OP*e*NDAP server data made available by OCEAN.US sources.

### 2.2 Standardized Database Readers

To simplify the modeler's work, the system provides a set of standardized database readers. These will be the interface into the Data Bank. It is important that these be the only method modelers use to grab data for model runs. Maintenance of the data links will thus not be a modeler's responsibility and can be handed off to the specialist who maintains the COMF readers. The most common failure mode of real time modeling systems is an interruption to data feeds at the source, usually caused by internal changes to operations, such as new data file formats or password authorization. The gage failure is also very common. With COMF, such failures are addressed in only one location, the COMF data base reader, and not on a model-by-model basis. The labor savings from this alone would justify the usage of COMF.

By forcing all models to use these readers, all the raw input data files for all models are of similar type. This is necessary to allow model maintenance and simple switching from one input data source to another different source, such as is needed when a meteorological station is taken down for repairs, or a river stage gage is removed from service. Output of the standardized readers is only a few types of files. Data are only time series at stations and time series of 2D fields. Most station time series are simple ASCII files consisting of only date/time and observation. The two (or three)

dimensional field data will use the CSDL standardized HYDRONetCDF file formats (Gross and Lin in preparation.). Some station observation data can be placed into NetCDF time series files.

### 2.3 CSDL Modelers Library

The standardized input and output files are supported by a growing list of routines used to read and output them. A library is maintained containing these subroutines. Model interfaces should be based on the library whenever possible. The library should not allow special case programs designed for one particular modeling method, estuary, Great Lakes, or coastal region. However general interface programs to POM, ROMS or other off the shelf models will be greatly encouraged.

### 2.4 Standardized Outputs

The NOAA CSDL/CO-OPS model outputs must present uniform format to intramural and extramural partners and stakeholders. All of the operational forecast system models must output similar files in order to allow 3rd party software to access our model results. Ocean US has declared that DODS (or similar technology) will be used to serve model results to the world using the extremely flexible NetCDF format (Refer to DMAC, http://dmac.ocean.us/dacsc/imp_plan.jsp). Part of the COMF project has been the development of a standard NetCDF output format for oceanographic models which adheres to all meta-data standards expected by Ocean US.

#### 2.4.1 Graphics

A suite of graphics programs has been created. They are designed to read only the NetCDF standardized output files. The first level of graphics programs creates time series and field graphics for dissemination on the CO-OPS web pages. A generalized script, GRAPHICS.sh (Appendix B 17), calls IDL programs tailored to the graphics requirements of the CO-OPS Web Services. This script uses the NetCDF output of the models and accesses NWLON databases for the comparison observation data.

The NetCDF standard will allow reusable graphics software to be created to serve all CSDL and CO-OPS modelers. Improved products, including graphics is a recognized priority that will be implemented incrementally with new versions of COMF. Example improvements may include integration with other CO-OPS products and web sites with capability to zoom and capability to link to station time series from field plots. Future improvements of graphics should include animations, 3D, "Slice and Dice 3D", and GIS converters. GUI interfaces to provide a simple way to view model results in detail would also be useful to the modelers.

#### 2.4.2 Web Pages

CO-OPS web pages are generated from the standard model output files. It is important to be able to quickly and reproducibly put up a NOAA web page with a new model. By standardizing the graphics and web pages, we can build a "new" web page in a matter of days, rather than months.

Enhanced capabilities, such as on demand drogue tracking, can be added to existing models and web pages without extensive retooling.

### 2.4.3 CORMS

The CORMS (Gill, Stoney, and Bethem 1997) will be used by all CO-OPS models to assure that the models are continuously running and to notify all necessary personnel when problems do occur. The CORMS reporting tool is a web based interface with a series of status flags indicating the operational status of hardware, data links, model runs and data quality. The COMF automatically produces these flags and will allow all of our models to easily integrate with the existing CORMS tools.

### 2.4.4 Model Skill Assessment Tool

Part of the process of creating a NOS operational model is the application of standard procedures for evaluating the accuracy of models (NOS 1999; Hess et al. 2003). Certified software to calculate the statistics have been developed which ingest COMF standard output files. The skill assessment software is described in a NOAA technical report (Zhang, in preparation).

## 3. DATA ACCESS METHODS

### 3.1 Readers and Variables

The most important functional unit of the system is the collection of data access tools. The directory in which these reside is the Operational Quality Control System (OQCS), the data access process, which also applies QC tests to incoming data. The tools are used to access the data bank in a standardized manner to provide for the modeler all the external real-time data and forecast results which will be used by the models. Tools are provided to query the databases with a starting and ending time and variable type. The tool will then give back, usually, simple ASCII text files with time series of quality controlled and gap filled data. There is quite a bit of programming going on behind the scenes to output these data, but the hope is that the modeler will have no need to worry about that and will only need to be familiar with the front end scripts.

#### 3.1.1 Readers and Variables

**Table 1** Access database.

| Variable | Script Name | NWLON[11] | ETSS[22] | NDBC[33] | NAM[44] | USGS[55] |
|---|---|---|---|---|---|---|
| Water Level | WLQCF.sh[a] | TS3 | TS3 | - | - | TS3 |
| Wind | WINDQCF.sh[b] | TS2 | - | TS2 | CDF,TS2 | TS2 |
| Air/Water Temperature | TEMPQCF.sh[c] | TS1 | - | TS1 | CDF windqcf | TS1 |
| Pressure | PRESQCF.sh[d] | TS1 | - | TS1 | CDF | TS1 |
| Salinity | SALTQCF.sh[e] | TS1 | - | TS1 | - | - |
| Currents | CURRQCF.sh[f] | TS2 | - | TS2 | - | - |
| Discharge | RIVERQCF.sh[g] | - | - | - | - | TS1 |
| Stage | WLQCF.sh[a] | - | - | - | - | TS3 |

[11] NWLON : National Water Level Observation Network, including PORTS.
[22] ETSS : Extra-Tropical Storm Surge model.
[33] NDBC : National Data Buoy Center's Buoys and C-MAN stations.
[44] NAM : North American Mesoscale numerical weather prediction model.
[55] USGS: US Geological Survey streamflow data.

[a] Appendix B 53 `Script Name: WLQCF.sh` : Appendix B 53
[b] WINDQCF.sh : Appendix B 49
[c] TEMPQCF.sh : Appendix B 45
[d] PRESQCF.sh : Appendix B 33
[e] SALTQCF.sh : Appendix B 43
[f] CURRQCF.sh : Appendix B 9
[g] RIVERQCF.sh : Appendix B 40

Table-1 shows the available variables, the script which controls reading the variable, the available databases, and the availability of that variable inside the database and the type of output file. Output files are ASCII time series with 1, 2 or 3 data entries, abbreviated as: TS1 (time series with 1 data

entry), TS2 and TS3. NetCDF files are indicated as CDF. They will contain fields of data or perhaps a number of scatter points built up into time series. The dashes indicate data are not a part of that database. Some data is not available at all stations within a database.

The OQCS scripts are extensively documented in the Appendix B. Table-2 is a list of the most commonly used scripts. Each of these "front end" scripts contains calls to particular scripts written for each individual database, and sometimes to a single database for a single type of data. But those details should not be noticed by the modeler, who should see nearly identical functionality across databases and across most types of data.

Table 2 Data grabber scripts description.

| | |
|---|---|
| WLQCF.sh | Returns observed, astronomical predicted tide and non-tidal water level, and stage. |
| CURRQCF.sh | Returns observed water velocity, U Eastward, V Northward. |
| SALTQCF.sh | Returns observed salinity. |
| TEMPQCF.sh | Returns temperature (top, bottom or air temperature). |
| WINDQCF.sh | Returns the wind velocity, U Eastward, V Northward. |
| PRESQCF.sh | Returns the atmospheric pressure. |
| RIVERQCF.sh | Returns the river discharge, or stage. |

All readers are called in a very similar manner. The region station ID, database name, starting time, ending time, time sampling frequency and output file name. There are a few exceptions noted below for each QCF.sh script.

The input to these scripts is

**QCF.sh $sid $database [Qualifier] "$t1" "$t2" $DT $outputfilename*

Table 3 *QCF.sh calling parameters.

| | |
|---|---|
| $sid | Station ID. The NWLON station IDs or sometimes the NDBC station IDs. |
| $database | The same data might be found on different databases. For instance, wind data can be obtained from NWLON PORTS stations or the NDBC buoys. The choices are listed and discussed below. |
| [Qualifier] optional | Some of the scripts include a qualifier to further refine the measurement type. For instance temperature can be water temp., WT, or atmospheric temp., AT. |
| $t1 | The starting time. A string with space delimited integers, as YYYY MM DD HH MIN. |
| $t2 | The ending time. A string like $t1, Both of them need " " to protect the spaces. |
| $DT | The time interval for the output time series (in hours). For instance, if six minutes data is desired, DT=0.10. |
| $outputfilename | String for the output file name. |

An example of calling WLQCF.sh using sh shell variables:

*sid=8638863*
*database=NWLON*

*t1="2003 02 15 12 30"*
*t2="2003 02 16 18 36"*
*DT=0.10*
*outputfilename=CBBTWL.DAT*
*WLQCF.sh $sid $database "$t1" "$t2" $DT $outputfilename*

or

*WLQCF.sh 8638863 NWLON "2003 02 15 12 30" "2003 02 16 18 36" 0.10 CBBTWL.DAT*

The temperature script allows for air temperature or water temperature by adding the extra qualifier variable of type after the database:

*TEMPQCF.sh $sid $database AT "$t1" "$t2" $DT airtemp.dat*
*TEMPQCF.sh $sid $database WT "$t1" "$t2" $DT watertemp.dat*

For some stations, there is access to surface or bottom water temperature with:

*TEMPQCF.sh $sid $database WTS "$t1" "$t2" $DT watertempsurface.dat*
*TEMPQCF.sh $sid $database WTB "$t1" "$t2" $DT watertempbottom.dat*

### 3.1.2 Time Series Output files: TS1, TS2, and TS3

Outputs for these routines are ASCII files with date time and the data field:

yyyy mm dd hh mm fh data

fh is the forecast hour. For observations this is always equal to 0. For forecasts the value may span 0 to e.g. 48 or however long the forecast is for. The valid time for the data is simply the yyyy mm dd hh mm on the record line. DO NOT add the forecast hour to this value.

The water level has three data fields: observed water level, tidal predicted and non-tidal. The current and wind files have two data fields: U (eastward) and V (northward). Since quality control and gap filling have to be performed in Cartesian coordinates we will not provide a speed and direction output. We also feel that the ambiguity of direction as flow towards and flow from, which differentiates currents and winds, is not worth resolving, hence no angular direction field.

These tools are all found in the /oqcs directory where most of them are found as shell scripts in /oqcs/scripts (Appendix B). The UNIX path variable must be set to access this directory and /oqcs/bin. $PATH is set in setenvironmentvariables.sh (see Module 0, Appendix B 41).

### 3.1.3 Ramp Filling for Water Level

Most hydrodynamic models will behave very badly or crash if the first outer boundary condition for the water level does not exactly follow the water level used on the previous time step. When a model is started with a hotstart file, a mismatch in the outer boundary water level can easily occur. Where the observation data is complete and clean, there is seldom a problem. But if the previous run had missing data at the end of the run, the model may have been executed using persistence or some other artificial forward filling technique. The next run access of the database may have found

the previously missing data replaced by good data, or it may use some other sort of back filling. A special function is available through WLQCF.sh which prevents discontinuous water levels.

The solution to this problem adjusts the forcing data to match the hotstart data. The adjustment is done by adding an offset to the non-tidal portion of the water level signal. The adjustment offset is then continued, but slowly reduced to zero. The time to reduce the adjustment to zero is called the ramp time. WLQCF.sh can apply this ramp adjustment to the data automatically. If a single water level value or a previous time series file is given to the WLQCF.sh command line, the output time series will be adjusted at the start time and a six hour ramp is used to reduce the adjustment. The water level at the start time maybe provided:

*WLQCF.sh 8638863 NWLON "2004 01 12 12 00" * [4]
*"2004 01 13 00 00" 0.10 CBBTWL.DAT 0.543*

This will cause the output file to be adjusted to 0.543 at the start time 2004 01 12 12 00. Any adjustment value which was added will be reduced to zero over 6 hours. Or the water level valid at the start time can be extracted from a previous time series file:

*WLQCF.sh 8638863 NWLON "2004 01 12 12 00" *
*"2004 01 13 00 00" 0.10 CBBTWL.DAT CBBTWL.LASTRUN.DAT*

This reads the second data file, *CBBT LASTRUN.DAT*, for the water level at the start time.

A graphical description of the process is presented in Figure 1. In the example, nowcasts are run every 12 hours. The observation data available for Nowcast 1 from Jan 12, 00:00 to Jan 12, 12:00 are missing from 9:00 to 15:00. Therefore, the WLQCF.sh script must persist the 9:00 non-tidal value until 12:00 (red line). Nowcast 2 from Jan 12, 12:00 to Jan 12, 24:00 must start with the previous nowcast HOTSTART file which specifies the water level value at Jan 12, 12:00. WLQCF.sh uses this value and linearly fills to the non-tidal offset six hour later at Jan 12, 18:00 when upon the full observation is used (green circles). The interpolations and gap filling are all done only to the non-tidal water levels. The full data are obtained by simply adding back the astronomical prediction values for the time period.

The forecast water levels from the Extra Tropical Storm Surge forecasts usually have fairly large offsets from the last observed water level value. The WLQCF.sh script applies the same offset correction method to the ETSS data, but it uses a very long ramp time of six thousand hours, effectively persisting the last observation correction value for the full 36 hours of the ETSS forecast.

The default ramp length for ETSS is six thousand hours and for all other data sources is six hours. However the ramp length can be adjusted. An eighth parameter is given to the WLQCF.sh call to specify the ramp length (e.g. 16 hours in the following example):

*WLQCF.sh 8638863 NWLON "2004 01 12 12 00" *
*"2004 01 13 00 00" 0.10 CBBTWL.DAT CBBTWL.LASTRUN.DAT 16*

[4] Back slash is a line continuation in Shell script language.

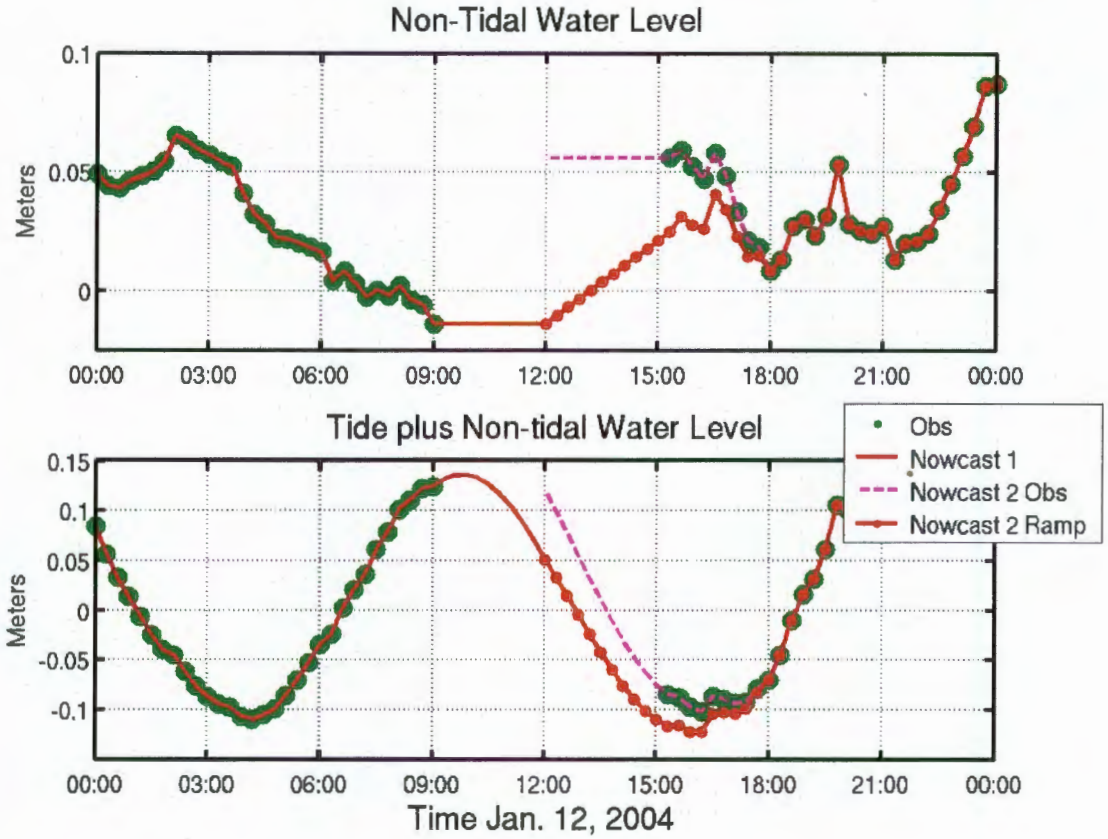



Figure 1 WLQCF.sh Missing Data Ramp Method.

### 3.1.4 Forecast Wind Field Access

The forecast wind data, such as the NCEP NAM model forecast guidance are accessed using the same scripts as above. Except that the output file may be quite different. The wind forecast output file is a NetCDF file. The station ID for a field forecast is the minimum and maximum latitudes and longitudes of the box in which the desired data will be found. No re-sampling in time is done to the NetCDF files so the DT is a dummy. The output file is the full duration of the closest forecast file which contains the starting time. Thus the end time is also a dummy.

An example of calling the NAM wind:

*WINDQCF.sh "-78 -74 36 40" NAM "$time_now" "$time_forecastend" 1.0 windsnam.nc*

The NetCDF file was created by the ODAAS system by reading and translating the original NCEP NAM model GRIB output file. The velocity is rotated to Eastward and Northward components. Time and other structures in the NetCDF file meets with the HYDRONetCDF standards. The NAM NetCDF file also contains air temperature, relative humidity, short- and long-wave radiation fluxes, total precipitation, cloud cover, mean-sea level pressure, and surface pressure.

### 3.1.5 Forecast Station Wind Reader: NAMSTATION

The single station output file is also available for WINDQCF.sh using the NAM forecast.

```
WINDQCF.sh "-78.543 36.443" NAMSTATION "$time_now" \
  "$time_forecastend" 1.0 windsnam.t2
```

This will interpolate the NAM forecast fields to the longitude, latitude specified. Output is in the TS2 format for uwind and vwind data. However an additional file is also saved which is a single station NetCDF with all of the NAM forecasts at that one station. It will be named by appending the NetCDF extension ".nc" to the end of the requested filename. In the above case two files will be created: windsNAM.t2 and windsNAM.t2.nc The NetCDF file uses just the starting time and ignores the delta time specification. However, the TS2 file returned is extrapolated, gap filled and interpolated to the specified time spacing (1.0 hourly in the example above). The same functionality to return a NetCDF file has been added to TEMPQCF.sh and PRESQCF.sh to use NAMSTATION.

### 3.2 CORMS Flags

Continuous Operational Real-Time Monitoring System (CORMS), requires information about the status of each raw data access. The operational model will send to CORMS a file containing lines describing the success of the various actions which make up a successful model run. These flags are interpreted and displayed to the CORMS operators as red, yellow or green indicators of the health of the system, (e.g. The CBOFS flags, Figure 2). A CORMS flag must be provided by the COMF data acquisition scripts to indicate the percentage of good data returned for each data access attempt. Each raw data access will be checked for number of lines returned and number expected. A script is provided which does this function, cormspercent.sh. Given a raw data file, the starting and ending times and the expected delta time of the database used, this script prints out the percentage of good data received. It will be called from inside the *QCF.sh scripts. For instance these lines are inside TEMPQCF.sh

```
deltat=0.10
CORMSPERCENT=`cormspercent.sh "$tstart" "$tend" $deltat $TDAT`
echo "TEMP_"$sensor" "$stationid" "$CORMSPERCENT >>  $CORMSLOG
```

The result of each call to a QCF.sh script will be the concatenation to $CORMSLOG of another CORMS flag line indicating the result of a database read. The very special environment variable $CORMSLOG is the name of the file in which all CORMS flags will be recorded. By default it is set to /dev/null by the setenvironmentvariables.sh script, specified in Module 0. At the top of the model run script this variable should be set to a real filename and exported. A first line should be written to it to initialize and perhaps provide information for CORMS flag processing. At the beginning of the model run script (e.g. MAIN_CBOFS.sh) these lines should be included:

```
source /COMF/oqcs/setenvironmentvariables.sh
export MODELDIR=/COMF/ohms/cbofs
export CORMSLOG=$MODELDIR/execlog/corms_raw.txt
time_now=`date -U`
```

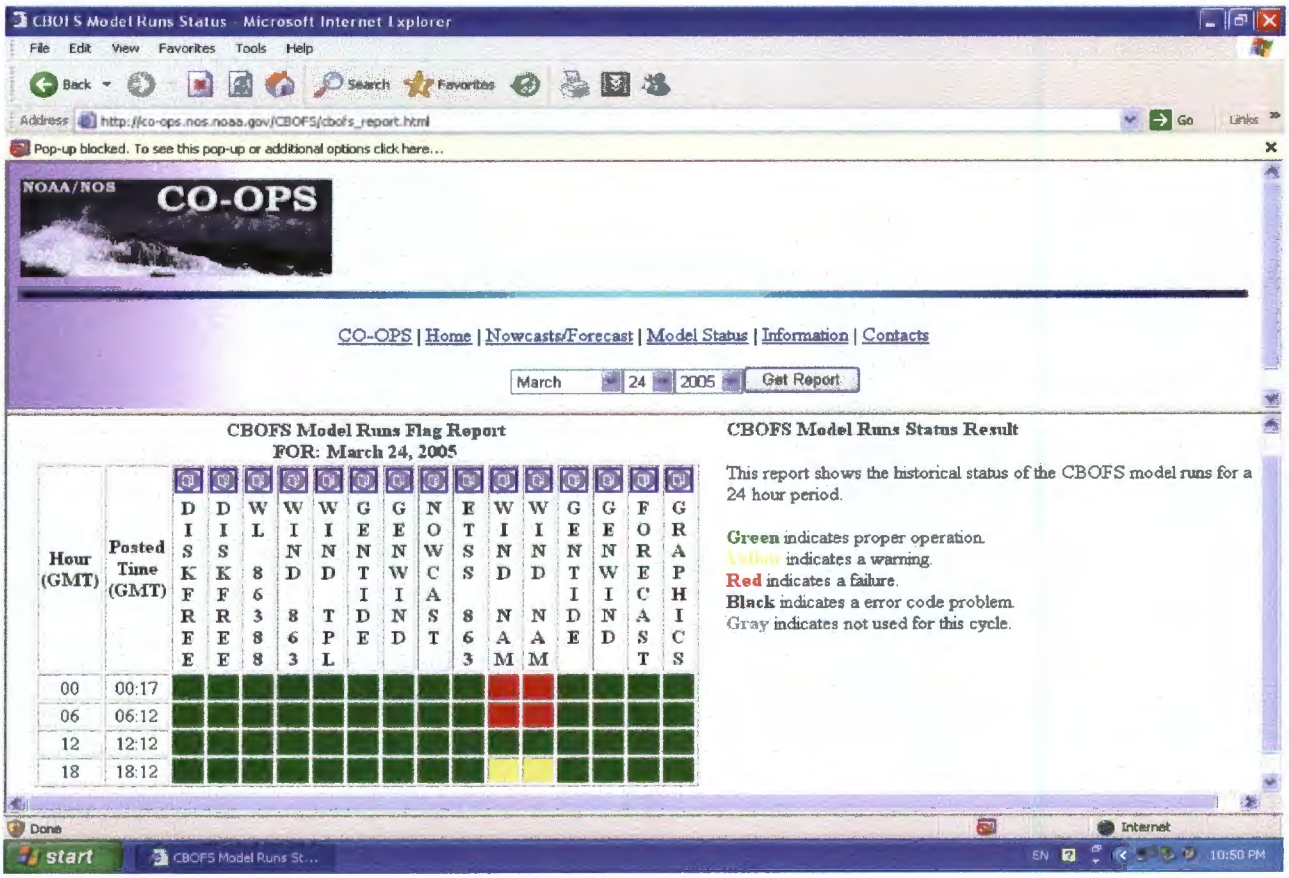
*echo "CORMS flags for "$MODELDIR" "$time_now > $CORMSLOG*

Subsequent calls of the QCF.sh scripts will populate the $CORMSLOG file with entries like:

TEMP_AT 8638863 87.5
WL 8638863 95.6
WIND 8638863 66.7

The $CORMSLOG will be processed at the end of a model run, in MODULE 8, to create the information needed by the CORMS web page displays. The script MAKECORMSFLAGS.sh (Appendix B 23) is provided to perform this function. It calls CORMSPROCESS.pl which accesses the specially named file, $MODELINFO/corms_table.txt, which contains the expected CORMS flags and their threshold values for the model. The new CORMS flags are processed and the red, yellow and green flags are put in a file for the CORMS web site. Finally that flag file is copied to the CORMS web directory, $MODELWWW, where it is displayed to the web page, CBOFS CORMS flags. (Figure 2) In the web page, the complete name of flags will appear when one clicks the symbol of each column.

Figure 2 Sample of CORMS flags web page for an OFS.



### 3.3 Data Banks

A variety of databases are interfaced with the QCF.sh scripts. For each of these databases, there are several "middleware" programs which allow the particular access method and format problems of the database to be solved and hidden from the COMF user. Some of the access and format solutions for the databases may change through time. For instance the computer IP addresses, or web access protocols change about every other year for most of these. The COMF system's greatest contribution to model stability is to isolate these problems to a single interface program, which can be relatively easily fixed.

#### 3.3.1 NWLON

The National Water Level Observation Network (NWLON), database includes observations, the water level gages and many meteorological stations used by most models. The NWLON and PORTS data are combined and maintained by the CO-OPS (Bethem 1998; Burton 2000). Access is given through the CO-OPS script get_data_nwlon_db.sh (Appendix B 11) which directly accesses the NWLON SYBASE SQL database. It returns the date, time, observation and QC flags from the database. The *QCF.sh script which calls this uses the QC flags to reject bad data records. This access method can get PORTS PUFFF data which is only about six-12 minutes old (Evans, French and Bethem 1997). get_data_nwlon_db.sh is also the interface to the astro tide predictions provided by the program pred_ngofs.f (Appendix C 20). This is the source of the tidal predictions provided in the TS3 files from WLQCF.sh .

#### 3.3.2 NWLONweb

An earlier version of the COMF script accessed NOS' WLON database through the CO-OPS web page. Rather than go directly to the SYBASE SQL database, this runs off their general purpose CGI script which is accessed through an HTML call with wget. This is more complicated and, at the same time, simpler than the get_data_nwlondb.sh. However it is susceptible to arbitrary changes in the CO-OPS web pages while the ISQL method is not. It has the advantage that it works on any computer with Internet access, not just the computers inside the NOAA firewall, as required by get_data_nwlon_db.sh.

#### 3.3.3 NDBC

This also uses an HTML wget command to access the raw data files off the National Data Buoy Center (NDBC) web site. The data files are usually only the last 30 days. Old data is not available. However several years old data is downloadable from their archives. An alternative method to be called NDBCarchive is being developed to aid the modeler in doing retrospective runs.

#### 3.3.4 USGS

The United States Geological Survey (USGS) stream gage data provides the fresh water sources for the estuarine models. The USGS web pages are accessed using the wget command. The files returned are of variable format so a parsing program was written to draw the various data types out of their files.

### 3.3.5 ETSS

The Extra Tropical Storm Surge (ETSS) forecast model results are held in ODAAS. The ETSS data is given at locations specified by mnemonic codes, e.g. cb, ny. The reader uses the NWLON station id most appropriate to the ETSS station. This is necessary to access the appropriate tidal constituent data which only exists at NWLON station.

### 3.3.6 NAM

The large GRIB files holding all of an NCEP NAM model forcast cycle are downloaded by ODAAS 4 times per day. They are then converted to NetCDF format by an ODAAS NCL script. The files are purged weekly, thus only the most recent files are available. The NCLwindgetNAMsub.sh script subsets the large NetCDF file for just surface winds, pressure and temperature within a specified longitude, latitude boundary. The NetCDF also contains the time and the longitude and latitudes of the data.

### 3.3.7 Other NCEP models

In the future, COMF scripts will be modified or new ones written to handle analyses and forecasts from NCEP other operational meteorological forecast systems, such the Global Forecast System (GFS), the Weather Research and Forecast (WRF) model, and Rapid Update Cycle (RUC).

## 4. SYSTEM DESCRIPTION

The COMF will be described by working through all the procedures necessary to make a standardized hydrodynamic model run. The process of creating an operational model consists of many steps and decisions. First a geographic region is identified and the major physical parameters of interest to the public and necessary to the model are identified. A numerical hydrodynamic model capable of working in the area is identified. The geometry, numerical grid and bathymetry are collected or constructed for the region. Model initial conditions are specified from a previous run of hotstart of observed data. Appropriate external forcing requirements are identified. This is the point at which available data, real time data and external data bases are identified. Methods for gathering those data are created. Non-operational tests are done on the model to develop the code and accuracy. These non-operational tests may include: 1) hindcast on demand; 2) astronomical tidal simulation; 3) one year hindcast; 4) one year of repeat daily nowcast/forecast cycles. COMF can be used to easily run these alternative scenarios. Finally all these ingredients are combined in an operational system with web site and CORMS quality control added. The COMF dictates this last step explicitly, but familiarity and usage of the COMF tools can greatly aid the initial design and configuration.

This document is a broad overview and more thorough description of the design and details of all parts of COMF. A step by step user guide to setting up a COMF-based model forecast system is provided in Appendix A as "Build a Model with COMF".

An operational model consists of a program which executes the ten modules in sequence. The method of running the model is via a crontab'ed shell script which executes the ten modules. The daily run schedule of the model will be specified by its CRONTAB file (e.g.):

*10 0,6,12,18 * * * /COMF/ohms/CBOFS/scripts/MAIN_CBOFS.sh*

The MAIN_CBOFS.sh script is referenced throughout this document as an example of a COMF-based system and the module definitions. CBOFS is the Chesapeake Bay Operational Forecast System which was the first CSDL/CO-OPS model system to become operational (Gross, Bosley and Hess 2000, Gross 2002). It has subsequently been retro-fitted into COMF. As with all real world examples we immediately see an exception to the idealized system. The model is run in both nowcast and forecast cycles. To separate the two, and yet keep the structure, the modules 3, 4, and 5 are repeated for the forecast after one pass through the nowcast. Forecasts are not always done so these modules are specified inside an IF clause.

### 4.1 Directory Structure

COMF provides a very logical directory structure for operational forecast system development, operations and maintenance. The standardized components discussed above (data tanks, data grabbers etc.) are provided in centralized directories and are accessible by all operational forecast systems. The directory structure is described in Table 4.

Table 4 COMF-based system directories description.

| Directory Name | Meaning of Name | Purpose |
|---|---|---|
| /comf/odaas | Operational Data Acquisition and Archiving System | Data banks |
| /comf/oqcs | Operational Quality Control System | Data readers and processors |
| /comf/oqcstools | Operational Quality Control System Tools | Miscellaneous software tools |
| /comf/opds | Operational Product Dissemination System | Graphics codes |
| /comf/ohms | Operational Hydrodynamic Model Systems | OFS (models) specific codes and scripts |
| /comf/archive | Archive | archive of OFS output |

The individual Operational Forecast Systems will reside beneath ohms, e.g. ohms/**OFS, where ** is replaced with the 2 letter abbreviation for the water body (e.g. cb for Chesapeake Bay yields CBOFS; le for Lake Erie yields LEOFS; etc.)

The first COMF requirement of each model is a standardized directory structure. Models are designed to run from a crontab shell script. The crontab'ed script and all other files it refers to reside in the standardized directory and file system. Computer resources required by the models are all provided by the COMF directories which also adhere to the standardized directory system. The system can be visualized as several directories containing the different parts of the system. The highest directories are the functional groups: The data bank: ODAAS. The quality control and data access programs: oqcs. The additional high level tools like NCL and SYBASE clients: oqctools. The output and post processing programs: opds. The collection of operational hydrodynamic models: ohms. The individual models, assuming there is more than one on the same disk system, will reside beneath ohms, e.g. ohms/CBOFS.

The directories of the system will all have a similar structure. For example under the CBOFS model directory, /COMF/ohms/CBOFS, will be found the directories listed in Table-5

Table 5 Subdirectories of Operational Forecast System or the OQCS.

| | |
|---|---|
| ./scripts | Interpreted language code: Perl programs, shell scripts, and crontab scripts. |
| ./sorc | Compile language code: source code files for Fortran and C program. |
| ./bin [Linux][sgi] | The executables created from the source files. Two (or more) directories are maintained to keep executables for different computer architectures separated. Makefiles for these will be found in the sorc directory. No executables should be found here which does not originate in the sorc directory. Also contains liboqcs.a, the collection of Fortran subroutines maintained for the system. |
| ./archive | Results files are stored here. These can be model output or databank data files. Subdirectories can be established with categories of output results. |
| ./info | Constant data files. These include unchangeable data files, such as mesh geometry, climatologically forcing files, and template files. |
| ./execlog | Log files created when scripts run are placed here. Usually overwritten, these files provide information on the most recent runs of models. CORMS flags will also be found here. |
| ./docs | Information documentation about the system. Copies of Tech reports and other handouts prepared for the model. Description of the software utilities. Static HTML pages describing the system. Change logs where the modeler will record any changes to system made after the system is declared "operational". |
| ./work | Where the model is run. This directory contains expendable files created during the execution of the system. Usually the first action of the model is to change directory to ./work where any files operation will land. Output files are written here and subsequently written to archives after a full model run completes. |
| ./init | All hotstart and other initialization files are stored here. |
| ./wwwgraphics | All graphics produced for the web are moved here from the work directory after generation. |

### 4.2 Module 0: Environment Variables

Key of much of this new system is a set of environment variables which describe the location and path to the directories and executables. The script /COMF/oqcs/setenvironmentvariables.sh sets up most of these variables. It also sets up some somewhat hidden functionality, such as the NCL and NCARG variables to enable the GRIB to NetCDF converters. Also the SYBASE SQL system for the reading of the NWLON data is initialized. Calling this script at the top of MAIN model script should set up all the rest of the run. To call this from the prompt and run interactively will require that your UNIX shell is either sh or bash (csh and other shells use incompatible syntax). All model scripts cover this by leading off with the defining line #!/bin/sh. The environment variables must be accessible to all scripts and programs, so the proper way to initialize them is with the source command:

*source /COMF/oqcs/setenvironmentvariables.sh*

After this sets up most of the variables, you will want to set the path to your own model. This usually means overwriting the MODELDIR variable and adding it to the path:

*export MODELDIR=/COMF/ohms/cbofs*
*export PATH=$MODELDIR/binlinux:$PATH*

In theory if all of your scripts make use of the exported execution $PATH and use $MODELDIR for specifying your model directories then the code and model should be totally relocatable. After tarring the main directories and placing them on the target machine then the setenvironmentvariables.sh script should be checked and altered for any differences between the computers. We are maintaining a number of these scripts, named after the individual machine to which they apply, e.g. setenvironmentvariables_bassmmap.sh, setenvironmentvariables_gbofs1.sh. No editing of scripts down inside your directories should be needed. Maybe the CRONTAB file needs the path to the setenvironmentvariables.sh script to be changed. Several CSDL and CO-OPS computers are represented with pre-made setenvironmentvariables.sh:

*setenvironmentvariables_bassmmap.sh*
*setenvironmentvariables_cbbay.sh*
*setenvironmentvariables_gbofs1.sh*
*setenvironmentvariables_linux.sh* *Used on both gbofs1 and gbofs2 machine*
*setenvironmentvariables_sgi.sh*
*setenvironmentvariables_dsofs1.sh*

### 4.3 Module 1: Computer System Tests

This module checks that the computer system is ready for the model run and produces CORMS flags indicating the status of the computer. Usually we are interested in whether the disk systems are working correctly and have sufficient empty space to run and save the model results. In addition the OFS_CONTROL.sh script (Appendix B 31) is executed. It prevents multiple, overlapping runs of the same model. This failure mode produces such confusing results that we don't simply flag it with a CORMS flag, but try to prevent it altogether.

### 4.4 Module 2: Model Timing

Model timing is determined only by the time of execution, i.e. the crontab time and the time variable in the previous simulations "hotstart" file. Models must determine the value of the time variable of the previous simulation's hotstart file (time_hotstart) and produce a nowcast to "now" (time_nowcastend). During a forecast cycle, a many hour forecast into the near future (time_forecastend) will be produced, the length of which is usually limited by the duration of the meteorology forecast models available (the example below will use 36 hours).

Previous models executed on complicated timing schedules dictated by the data collection and dissemination phase. Now the data collection and archive functions are removed from the model system. So the execution method of the model is independent of the data acquisition. The data query tools are designed to grab a certain time range of data, and if that time range comes up to the present or into the future the tools will extrapolate the data into the requested bounds. The nowcast

time is created from querying the date function (time_nowcastend=`date -u +"%Y %M %d %H %m"`). Note the use of the "-u" option to guarantee the use of UTC throughout the system. The start of the nowcast time is obtained by querying the model's hotstart file (e.g. hotstart_out.*) with a modeler supplied function (e.g. readhotstart_out.*). The end of the forecast time will be 36 hours after "now". Inside the run script, MAIN_**OFS.sh, will be something like:

*time_hotstart=`readhostart.x hotstart_in.dat `*
*time_nowcastend=`date -u +"%Y %m %d %H 0"` (typically rounded to the top of the hour)*
*time_forecastend=`datemath $time_nowcastend + 0 0 0 36 0`*

The beginning of the graphics window will be time_nowcastend minus (m) 24 (24) hours (h) and defined by the syntax

*time_nowcastendm24h=`datemath $time_nowcastend - 0 0 0 24 0`*

The above time variables should be sufficient for most models. However, if additional times are required they should be referenced against the time_nowcastend variable using plus (p), minus (m), hours (h) and days (d).

### 4.5 Module 3: Data Access Tools

The Data Access Module has only a few calls to the QCF.sh scripts described above. The direct access data reads are all that go in this section. Since most of the maintenance problems of an operational system crop up with the changes and failures of the data access methods, it is very important to isolate them from all other specialized running requirements of the model. In this section there will usually be calls to WLQCF.sh for non-tidal water level forcing, TEMPQCF.sh and WINDQCF.sh for nowcast or forecast weather forcing data, and RIVERQCF.sh for riverine inputs. Data for graphical comparison, not model forcing, should not be accessed in this module.

### 4.6 Module 4: Reformat Data

The various input data files are now available. However, they are probably not immediately ingestible by the hydrodynamic code. In this module the standardized data files are reformatted to each model's requirements. This second level of middleware is unavoidable as CSDL presently uses a large variety of hydrodynamic modeling methods and codes. At this point, it is up to the modeler to provide this software.

The reformatting of each data file into the required model format must be done in clear, logical, distinct steps by scripts or programs with self describing names like:

*reformat_wl_nowcast.sh (or .x, .pl etc.)*
*reformat_wind_nowcast.sh*
*reformat_wl_forecast.sh*
*reformat_wind_forecast.sh*

### 4.7 Module 5: Run the Hydrodynamic Model

This module runs the hydrodynamic code. The code for a model run is allowed to be as heavily adapted to the estuarine case at hand as necessary. Thus very little about this module can be made universal. With the notable exception of the output file formats which must be the standard NetCDF. A routine must be constructed by each modeler to produce this output standard.

The raw "hotstart" file from the nowcast run must be named:

*hotstart_out.dat* (or appropriate suffix)

This file is then copied to /init/hotstart_in.dat (or appropriate suffix) if the file size is correct, using the script

*hotstart_copy.sh* hotstart_out.dat hotstart_in.dat filesize

The hotstart_in.dat file is then stored in the /init directory for use in the next nowcast run or a following forecast run.

### 4.8 Module 6: Archive the Results

Archiving the data is also a standardized procedure. The method is to write files into the archive directory ($ARCHIVEDIR) and its subdirectories which are labeled by the dates. $ARCHIVEDIR = $MODELDIR/archive is usually a logical link to a separate disk system. By using the logical link we maintain a simple local directory. The script ARCHIVE.sh (Appendix B 1) searches the working directory for files with key suffixes, and writes them in turn to the archive directory:

Table 6 Input files of archive.

| Name | Directory Location | Description |
|---|---|---|
| *graphics*.tar | $MODELWORK/ | Tarred graphics |
| hotstartout* | $MODELWORK/ | Model hotstart file |
| fields*.nc, station*.nc | $MODELWORK/ | Model NetCDF file |
| modelinput*.tar | $MODELWORK/ | Tarred inputs |
| fields*.grb, stations*.grb | $MODELWORK/ | GRIB files |

Table 7 Output files of archive.

| Name | Directory Location | Description |
|---|---|---|
| YYYYMMDDHHMI_graphics.tar | $ARCHIVEDIR/graphics/YYYYMM/ | Archived graphics |
| YYYYMMDDHHMI_hotstartout* | $ARCHIVEDIR/hotstart/YYYYMM/ | Archived hotstart file |
| YYYYMMDDHHMI_*.nc | $ARCHIVEDIR/netcdf/YYYYMM/ | Archived NetCDF file |
| YYYYMMDDHHMI_modelinput*.tar | $ARCHIVEDIR/modelinput/YYYYMM/ Archived/ | Input files |
| YYYYMMDDHHMI_*.grb | $ARCHIVEDIR/grib/YYYYMM/ | GRIB files |

### 4.9 Module 7: Make the Graphics

There is no post processing products of the models which are unique to one particular bay or model. This draconian statement is necessary to avoid the wasteful duplicative effort put into building post processing programs and providing output files to the public which has been the hallmark of the past. To achieve this end we have dictated a single, but quite capable, output file format which all models will provide. These are the CSDL HYDRONetCDF standard files described at length in the accompanying Tech Report. These output files form the basis of all of our output products, from statistical analysis tools to web page design. A suite of tools has been built which rely upon these NetCDF files and a short control file to produce similar graphics across models. The PORTS model web pages are based upon these graphics and are easily adapted to new models. The goal is that very soon after a new model has output a HYDRONetCDF file, a new web page, fully populated with graphics, can be displayed.

The GRAPHICS.sh script (Appendix B 17) controls all of the graphics creation processing for the models. Each model is required to have produced the NetCDF station and field files for both the nowcast and forecasts. They are assumed to be loaded into the archive directory structure. The GRAPHICS.sh will access the files by date, so that it may be run in a post processing mode as easily as it is run as part of the main model system. In addition to the input data, the modeler must prepare control files, $MODELINFO/plot_timeseries_wl.ctl (Appendix B 19), $MODELINFO/plot_field.ctl (Appendix B 18). These files must be copied to the $MODELWORK directory for GRAPHICS.sh to find. They contain a list of stations to be plotted, labels, units to plot, scaling, size of plots and the parameters to plot at each station. Under the guidance of these control files observation data is downloaded to a station type NetCDF file. A special NWLON data grabber was used, other than WLQCF.sh, which flags bad data for plotting rather than editing and gap filling the files, as do the other QCF.sh scripts.

A suite of IDL programs are called by GRAPHICS.sh to produce the plots under the guidance of the control files. The field plots are calculated and written to .png files for the web pages. This step is computationally intensive, can take several minutes and should not be forgotten when designing the timing cycle of operational models.

ARCHIVE_GRAPHICS.sh (Appendix B 2) is called next to store just the graphics into the archive directory.

The final step of the graphics module is the transfer of the graphics images to the web page. This is achieved via a special directory $WWWDIR which is usually a logical link to a remotely mounted directory of the web server computer. Specialized CO-OPS software detects the presence of new files in these directories and reconstructs the web page appropriately.

### 4.10 Module 8: Make the CORMS Flags

The $CORMSLOG file is processed at the end of a model run, in MODULE 8, to create the information needed by the CORMS web page displays. The script MAKECORMSFLAGS.sh (Appendix B 23) is provided to perform this function. It calls CORMSPROCESS.pl (Appendix B

7) which accesses the specially named file, $MODELINFO/corms_table.txt which contains the expected CORMS flags and their threshold values for the model. The new CORMS flags are processed and the red, yellow green flags are put in a file for the CORMS web site. Finally that flag file is copied to the CORMS web directory, $MODELWWW.

### 4.11 Module 9: Purge old files

Purge.sh removes old files from past model runs. This script uses a control file, $MODELINFO/Purge.ctl, to customize how the old files will be purged. The lines in the control file correspond to individual "rm" commands. Each line has a row for directory, file, and day string. The directory string is name of the directory under which the script will search for files to remove. The directory name is based on the type of files that it contains, such as NetCDF, graphics, execlog, etc. The file string is based on the name of the files to remove. This string corresponds to the part of the filename following the date prefix, and it can include wildcards. The day string is the age in days after which the files will be removed.

### 4.12 Web Pages

The ARCHIVE.sh and the final step of the graphics module simple copy files to the web server. The software supporting the web page and the dynamic generation of the CORMS QC flags is the responsibility of CO-OPS web operations.

### 4.13 Skill Assessment

The skill assessment statistics are based on time series of model results under several different scenarios. The basis of these time series will be the HYDRONetCDF station files with the tide and obs NetCDF files produced by the NetCDFgetstations_nwlon_fast.sh program (Appendix B 28). The operational system will be producing a few station files each day which can be concatenated together to produce a lengthy, year long time series file. A tool has been provided to perform this function, concatnetcdf.sh. Skill assessment programs are described in Aijun Zhang's Skill Assessment Software (Zhang, in prep).

## 5. OUTPUT FILE STANDARDS

### 5.1 NetCDF standard output format

The models will each produce two types of NetCDF files (UCAR 2005). The Station file is a collection of the time series of a variety of parameters at a small number of locations. These usually correspond to the PORTS water level gauges or current meters installed in the bay. The other NetCDF file type is the Field file. These are much larger and contain the results of the model at every location in the model's grid. The fields can be 2D or 3D as required. Both types of NetCDF files contain Meta data describing the attributes of the data and information on the model, the run times and even contact information designating the responsible parties. If the Meta data is in some form inadequate to answer all questions about the model run, then additional meta data can be added to our CSDL Standards. Meta data conform rigorously to the CF2.0 conversions (Gregory 2003). These NetCDF meta data standards are designed to maintain compatibility with a suite of data viewers and the OCEAN.US DMAC approved OPeNDAP data distribution methods (IOOS 2005). The nice thing about NetCDF files is that they are wonderfully backward compatible with old files which contain less data. Most changes do not affect the reading and plotting programs designed for the older files.

### 5.2 Log files

Log files are the redirected output of the various controlling scripts. They contain a wide variety of diagnostic information describing each run of the model. Most are overwritten each run cycle. The following logfiles will be created

*%YYYY%mm%DD%HH%MMdiagnostics.log* ..... (Output from MAIN_**OFS.sh)
*nowcast_model.log* ..... (Output from the execution of the nowcast model run)
*forecast_model.log* ..... (Output from the execution of the forecast model run)
*graphics.log* ..... (Output from execution of the graphics script)

## 6. SCRIPTS AND PROGRAM DESCRIPTIONS

The scripts and programs which make up the COMF system are mostly found in the oqcs/scripts and oqcs/sorc directories. In oqcs/scripts are found the full collection of the data grabbers and the other utilities used to build the operational main crontab script. In the oqcs/sorc directory are found the source codes for the many FORTRAN and C programs which perform specialized tasks, such as the date and time manipulators dateformat (Appendix C 4) and datemath (Appendix C 6). The scripts and programs are explained and listed in appendices B and C. In addition the COMF web page has a multi-linked appendix for both systems. Each script and program has received a standardized header section explaining its origin, purpose, calling method and update history. Extra information is included by an explanation text section which is displayed, as well. The OQCS SCRIPTS page lists all the scripts by function as well as alphabetically. The LIBRARY PROGRAMS page lists all the stand alone function codes as well as the collection of FORTRAN and C subroutines which are made available to modeler for compilation linking.

### 6.1 Scripts Library

The Scripts Library is the collection of the shell scripts which make up COMF. Most of them are located in the OQCS directory /COMF/oqcs/scripts. The setenvironmentvariable.sh script must be run before any of these scripts to complete paths and directory definitions. Ancillary software such as NCL or SYBASE client is needed for some of the scripts. These are located in /COMF/oqcstools and are put into the path by setenvironmentvariable.sh.

The Scripts Library web page is a complete directory access to the scripts and includes short descriptions of the scripts. The scripts will either be accompanied by their header comment notes, or they will have extra descriptions as needed. The Scripts Library web page is maintained by a script which reads all of the contents of the /COMF/oqcs/scripts and converts the files into syntax highlighted HTML code for display on the web page. This allows the web pages to be updated accurately and conveniently. Most of this information is found in Appendix B. The continuously updated web page with cross links is at the NOS Intranet web site http://tampabay.ncd-tcn.noaa.gov/~tgross.

### 6.2 FORTRAN Library

A library of FORTRAN subroutines has been made available. This is a library of ancillary subroutines which interface with the COMF as well as other subroutines which may be of use to modelers. This is a community resource and it is hoped that many more subroutines will added to this library by the modelers of CSDL. Modelers should try to use this library and not make copies of the source files. If updates on the files are performed this library will be updated, and all affected models should be recompiled. Version drift and eventual incompatibility can occur if personal copies of the files are kept by the individual modelers.

The FORTRAN sources are all found in /COMF/oqcs/sorc/library. Most of information is found in Appendix C. The makefiles in that directory will build the library liboqcs.a. The steps to building it are:

```
source /COMF/oqcs/setenvironmentvariables_linux.sh
cd /COMF/oqcs/sorc/library
rm *.o
rm *.a
make -f Makefilelinux
mv liboqcs.a COMF/oqcs/binlinux/.
```

Perform similar steps for SGI, producing *binsgi/liboqcs.a.*

Subroutines found in the library are now:

*HYDRO_netcdfs_fem.f*
*HYDRO_netcdfs_grid.f*
*HYDRO_netcdfs_station.f*
*gregorian.f*
*interp1.f*
*julian.f*
*wl_read_HTh.f*
*wl_read_oqcs.f*

The HYDRO_netcdfs files are explained in the HYDRONetCDF web page (http://ccmp.chesapeake.org/HYDRONetCDF/HYDRONetCDF.html)
To use these subroutines the libraries and include paths should be put onto your compile line:

```
-I/usr/local/include -L/usr/local/lib
-L/COMF/oqcs/binlinux -loqcs –lnetcdf
```

The first two are needed to make sure you pick up all the NetCDF libraries, which are usually stored in /usr/local/include and /lib. It sometimes was found that order mattered in the -loqcs -lnetcdf. A full compilation line for the quoddy program which also includes its own library:

```
lf95 --nap --nchk --ng -O --npca --nsav --ntrace --wide --ml cdecl \
        -I/usr/local/include quoddy5_1.1_main.f libquodinit.a      \
        -o q511init.x -L/usr/local/lib -L/COMF/oqcs/binlinux -loqcs -lnetcdf
```

The full Makefile used by Quoddy is found in Appendix C 17 as an example.

## 7. CVS AND TESTING ENVIRONMENT

The operational models and COMF will on occasion require updates and other development operations. In a system where there are many programmers working on several machines, it is necessary to use a manage system which organizes software changes and updates. Without some sort of organization, debugging and software fixes are not applied uniformly and some people may end up working with old versions of software which another person on the team has long ago fixed. Updates and version control for the COMF system and the operational models which reside in the OHMS directory are all controlled by the Concurrent Versioning System (CVS) software. This system is available on all UNIX/LINUX computer installations. It keeps track of all changes to the software and allows a robust method to assure that everyone is working with the most up to date and verified copy of the projects. The CVS operates by maintaining a master copy of the project in a repository. The developers and users "check out" the most recent and up-to-date copy of the project whenever they need. Bug fixes or new code are reintroduced to the repository and made available to all users through a simple "update" command. Old versions of all programs are retained so that the system may be returned to a prior state at anytime. A repository has been made for the COMF system. Separate repositories also exist for each individual modeling system found in OHMS directory.

When testing changes to the COMF, it will be necessary to work on a parallel system where mistakes can be made without interrupting the publicly accessible model results. At present several computers are designated as "operational". They all share the same /comf directories via an NFS mount. In addition there is a designated development computer which will have multiple copies of the COMF and the operational models. Some of these copies will be designated as "staging" versions, which will be complete systems undergoing refinement and testing. The staging versions will be fully capable of being switched to as a backup in case of failure of one of the operational computers. Other copies of the system on the development computer will be designated "development" and will be under active changing and testing.

The directory structure on the development computer will consist of:

*/comf*
*/comf/operations/ (An NFS link to operational gbofs1:/ngofs/)*

*/comf/staging/*
*ohms*
*gbofs, cbofs, nyofs ( possibly crontab'ed, for multi-day tests)*
*oqcs*
*oqctools*

*/comf/development/COMFname (Each developer can keep and use his/her own copy)*
*ohms*
*gbofs, cbofs, nyofs, gbofs2, c3po, tampabayofs*
*(models undergoing development and testing)*
*oqcs*
*(New scripts are being tried out.)*

*oqctools*
*/archives (To be linked to the ohms/models/archive.)*

### 7.1 CVS, Concurrent Versioning System

Concurrent Versioning System (CVS), is a software system which coordinates many developers working on the same code project. CVS is applied for COMF to keep track of changes and to prevent version drift. Version drift is especially dangerous, as someone can fix a problem which might not be reflected in a copy of the system left somewhere else without being updated. CVS does a good job of controlling this.

Some CVS resources:

Open Source Development with CVS, 3rd Edition by Karl Fogel and Moshe Bar.
Web Site: http:// cvsbook.red-bean.com/cvsbook.html

There is a nice Tutorial from the CVS home page.
CVS Tutorial : http:// www.cvshome.org/docs/manual

The multiplicity of systems is to be controlled by the CVS. This software package keeps all the programs in a directory structure which allows control over multiple versions. A new copy of the COMF can be "checked out" of the CVS and installed anywhere. The developer can work with this copy and make changes without affecting the other developers. Any changes deemed successful can be easily reintroduced to the CVS for use by all other developers. Other developers need only type "cvs update" at their prompts and all the new developments since they downloaded the system will be integrated into their current copies. No one has any excuse to be working with an out of date or divergent copy of COMF. Another feature of CVS is "sticky tags" which allow the state of the system to be recorded. This is needed when a new distribution has been ready for operational status. All files marked with the new version tag can be updated to the operational system. Further work on the COMF can continue while all the operational models take advantage of the updated system. All of the COMF systems are CVS copies from a repository which should reside somewhere distinct and safe, on the developer computer with tape backups, CVSROOT=/comf/CVSPROJECTS.

Development on /comf/development

This is how a programmer/developer will use /comf/development. When a developer wants to change a COMF script, like WLQCF.sh, she/he would cvs download a full system to her/his development directory, e.g. /comf/development/COMFuser/ and test it there (cvs co COMF; mv COMF COMFuser). Changes to environment variables to relocate the system will need to be implemented. (Create oqcs/setenviormentvariables_dsofs_User.sh.) The developer then uses the system and makes her/his improvements to various codes. When the program has been successfully changed the developer does a "cvs commit" to put the changes into the repository.

Then, a change must be verified and tested. For this the developer continues to use his or her development installation of /comf/development/COMFuser. The individual models are also CVS'ed and can be installed into the developers own ohms. The developer should run tests proving that the changes are working.

Next, a slightly more certified test can be done in the mod_dev version. mod_dev is an account on the development computer to be used for final testing. Ask someone with mod_dev permissions to do the cvs update to /comf/development/COMFmod_dev. Crontabs are running on this system to do initial testing of changes. This step will prove that the system is an active COMF compliant and relocatable system, if it can be successfully setup using CVS by someone other than the original model developer.

The next step will be proving the new system on the /COMF/staging system. This is preparatory to a full installation to the operational system. It means that /COMF/staging is relatively static. After deciding that the results in /COMF/development are correct, the mod_dev developer will mark the cvs files with a version flag. Then mod_dev would cvs update /COMF/staging with that version. The model's crontabs are run for a week. If problems do occur the developer can use cvs version controls to move /COMF/staging back to previous operating state. The developer, much chagrined, should return to testing in the developer's own home directory.

If no problems occur, then we can do a cvs update to the real thing in /COMF/operations. This will probably not be done by the programmer himself, but by a committee of COMF controllers. Using cvs flags the state of the tested COMF was recorded as a version number. That version will be uploaded to /COMF/operations. This way we will have a precise record of the full state of the COMF which was successfully tested in the staging arena. The version flags also allow other developers to be updating the CVS repository even in the middle of one developer's testing schedule.

### 7.2 Working with CVS.

CVS keeps the files in a possibly remote computer directory. The path to the repository is kept in an environment variable CVSROOT. If the computer is remote set it with:

*export CVSROOT=*
*:ext:user_name@dsofs1.nos-tcn.noaa.gov:/comf/CVSPROJECTS/*
*export CVS_RSH=ssh*

Or if the repository is local to your computer (i.e. you are logged into dsofs1) use:

*export CVSROOT=/comf/CVSPROJECTS/*

Soon we should have anonymous checkout ability, but for now the user name must be replaced with a real user account name for dsofs1. The anonymous method should use:

*export CVSROOT=*

*:pserver:anonymous@dsofs1.nos-tcn.noaa.gov:/comf/CVSPROJECTS/*
*export CVS_RSH=ssh*

a). Start a new version of the project in your own directory:

*cd /comf/development*
*cvs co COMF*
*mv COMF COMFuser*

b). Bring down the newest version from server to working directory before any further modification.

*cvs update*

c). Usually do coding at local directory:

*edit WLQCF.sh*

After modification and local test:

*cvs commit*

If it is a good version, set a version tag:

*cvs tag -R TAGNAME*

ex.

*cvs tag -R TAG2004Nov29*
or
*cvs tag -R NewWLQCF*

*cvs update -r TAGNAME*

Above commands will give a sticky tag name to all files as one set. Under the same tag name, files may have different version numbers. And one file can have many different tag names for the same version number.

After updating the project with cvs update -r TAGNAME, CVS server will not accept new commits. The working copy with tag is like a static snapshot of a moment of history, CVS won't let you change history easily. So use

*cvs update -A*

to remove tag, and then commit new modification.
Give meaningful tag name for each working copy to make recalls easier, and control the quantity of tag usage, to avoid unnecessary inconvenience.

d). If it is ready to move to */COMF/Staging* for pre-operational testing

*cd /comf/staging*
*cvs update -r TAGNAME*
or
*cvs co -r TAGNAME COMF(ProjectName)*

Then test by using the active crontabs.

e). If it is ready to be operational,

*cd comf/operational*
*cvs update -r TAGNAME*
or
*cvs co -r TAGNAME COMF*

If there are some unclear bugs, and you wanted to go back to a previous good version

*cvs update -r PREVIOUS-GOOD-TAGNAME*
or
*cvs co -r TAGNAME COMF*

Please do NOT use command 'cvs update -A' in operational directory. It is allowed to jump from version to version, but can NOT commit to repository. This is NOT a development directory.

System Mirroring and Operational Backup:

In case of a computer failure on one of the operational model computers, the development computer will be pressed into service to take over the operational system. Most likely /COMF/staging will be ready at any time to simply be turned on with its copies of the affected models. If a testing of a new version of COMF was underway, it may be necessary to interrupt the testing and bring back into staging the known version number of the system which was affected. After the broken computer is fixed the system can be restarted with the version and the /COMF/staging given back to the developers for their testing.

## 8. CONCLUSIONS

This report gives an overview of the COMF, its purpose and most of the techniques required for its usage. The extensive Appendices provide short descriptions of the component scripts and programs which make it up. However, the most important attributes of COMF is that it can be maintained by changes plus additions. Thus the most important document describing COMF is the continuously updated on-line HTML Intranet site http://tampabay.ncd-tcn.noaa.gov/~tgross.

## ACKNOWLEGEMENTS

The Coastal Ocean Modeling Framework (COMF) has been developed in the Coast Survey Development Laboratory (CSDL) and the Center for Operational Oceanographic Products and Services (CO-OPS). The authors would like to acknowledge the support from the chief of CSDL's Marine Modeling and Analysis Programs, Dr. Frank Aikman. The authors respectfully acknowledge the assistance, support and cooperation of Dr. Aijun Zhang. The authors would like to thank the many people who have helped us with this project, especially Dr. Ed Myers. The authors received invaluable assistance and support from CO-OPS personnel, Mike Evans, Zhong Li and Greg Mott. The authors would also like to acknowledge all the previous CSDL forecasters and modelers, and thank all the ODAAS developers.

## REFERENCES


Bethem, T. D., 1998. Systems Development Plan National PORTS Database. Ocean Products and Services Division / Information Systems Branch. 20 pp.

Burton J., 2000. A NWS Guide to the Use of NWLON and PORTS Computer-Based Products. NOAA Technical Report NOS CO-OPS 026. 31 pp.

Evans, M., G. French and T. Bethem, 1997. Information Systems Branch PORTS Uniform Flat File Format (PUFFF). Oceanographic Products and Services Division / Information Systems Branch. 23 pp.

Gill, S., W. Stoney and T. Bethem, 1997. System Development Plan CORMS: Continuous Operational Real-Time Monitoring System. NOAA Technical Report NOS OES 014. 41 pp.

Gregory, J., 2003. The CF Metadata Standard. http://www.cgd.ucar.edu/cms/eaton/cf-metadata/clivar_article.pdf.

Gross, T. F., 2002. Chesapeake Bay Operational Forecast System: Skill Assessment for 2001 and Improvements. NOAA Technical Memorandum NOS CS 1. 39pp.

Gross, T. F., K. T. Bosley and K.W. Hess, 2000. The Chesapeake Bay Operational Forecast System (CBOFS): Technical Documentation. NOS Technical Report OCS/CO-OPS 1. 69pp.

Gross, T and H. Lin (in preparation) MMAP modelers Standard NetCDF Files.

Hess, K. W., T. F. Gross, R. A. Schmalz, J. G. W. Kelley, F. III Aikman, E. Wei, and M. S. Vincent, 2003. NOS Standards for Evaluating Operational Nowcast and Forecast Hydrodynamic Model Systems. NOAA Technical Report NOS CS 17. 48pp.

IOOS 2005. Data Management and Communications Plan for Research and Operational Integrated Ocean Observing Systems. http://dmac.ocean.us/dacsc/docs.jsp

Kelly, J.G.W., M.Westington, E. Wei, S. Maxwell, and A. Thomson, 2001. Description of the Operational Data Acquisition and Archive System (ODAAS) to Support the NOS Chesapeake Bay Operational Forecast System (CBOFS). NOAA Technical Report NOS CS 10. 45pp.

Nault, J., 2001. NWLON/DMS Quality Control Software (QC): Functional Requirements Document. NOAA Technical Report NOS CO-OPS 030. 21 pp.

NOS, 1999. NOS Procedures for developing and Implementing Operational Nowcast and Forecast Systems for PORTS. NOAA Technical Report NOS CO-OPS 020. 33 pp.

UCAR, 2005. NetCDF Documentation.
http://my.unidata.ucar.edu/contents/software/netcdf/docs/index.html

Westington, M and J. G. W. Kelley, 2003. ODAA's Real-time River Information Acquisition for NOS Estuarine Forecast Systems in the Middle-Atlantic Region. NOAA Technical Report NOS CS 16. 46pp.

Zhang, A. and et al. (in preparation) The Skill Assessment Software.

## APPENDIX A. BUILD A MODEL WITH COMF

### TABLE OF CONTENTS

Appendix A 1 INTRODUCTION ..... 40

Appendix A 2 PREPARING THE MODEL AND DIRECTORIES ..... 40

Appendix A 3 MODELCRONRUN.SH: THE CONTROLLING CRONTAB SCRIPT ..... 40
Appendix A 3.1 MODULE 0: Set Up Environment Variables for Directories ..... 41
Appendix A 3. 2 MODULE 1: Computer System Tests ..... 42
Appendix A 3. 3 MODULE 2: Create the start and stop times ..... 42
Appendix A 3. 4 MODULE 3: Get data ..... 43
Appendix A 3. 5 MODULE 4: Reformat data ..... 43
Appendix A 3. 6 MODULE 5: Run the hydrodynamic model ..... 44
Appendix A 3. 7 FORECAST: Forecast Cycle Repeats 2-5 ..... 45
Appendix A 3. 8 MODULE 6: Archive the data ..... 45
Appendix A 3. 9 MODULE 7: Create the graphics ..... 46
Appendix A 3. 10 MODULE 8: Create the CORMS FLAGS ..... 47
Appendix A 3. 11 MODULE 9: Purge old files ..... 47

Appendix A 4 TESTING A NEW SCRIPT ..... 48

Appendix A 5 MOVING THE SYSTEM BETWEEN COMPUTERS ..... 49

Appendix A 6 CONCLUSIONS ..... 49

## Appendix A 1 INTRODUCTION

This guide to building a model in COMF will show the modeler how to set up the scripts which control the execution of the model. The customized setup will be achieved largely by altering the mainline Crontab script, MODELCRONRUN.sh (Appendix B 24), which controls the timing of the model through the UNIX crontab job control system. The mainline Crontab script consists of the ten modules described in the main manual and enumerated below. Each module represents a section of script code in the mainline Crontab script. Each section below tells what changes will need to be performed on MODELCRONRUN.sh to customize it for the new model. There should not be very many places in a MODELCRONRUN.sh which require changes.

An example MODELCRONRUN.sh script is appended to help guide the discussion. The text will refer to this script and point out where this script needs modification for new modeling systems. The web version of this document has links into an html coded version of the script. Download the file MODELCRONRUN.sh to have a copy which can be edited and adapted. (http://tampabay.ncd-tcn.noaa.gov/~tgross/sorc/MODELCRONRUN.sh). It is advisable that the user downloads a copy of this file and opens it up in an adjacent editor while working through this model building guide.

All scripts are written in the Bourne (sh) UNIX shell. The sh shell is available on nearly all UNIX and Linux computers, thus sh scripts are very portable. The Linux bash and ksh shells are backward compatible with sh, so sh scripts and commands will usually work in these shells but some advanced features of these shells won't work in sh. However csh is not compatible with sh. In order to avoid incompatibility problems, every shell script, including the MODELCRONRUN.sh, must lead off with:

*#!/bin/sh*

## Appendix A 2 PREPARING THE MODEL AND DIRECTORIES

The COMF system depends upon a consistent directory structure. These are described in general and detail in the main document, Directory Structure. The parent directory of a system is usually installed in a directory with a name like "/COMF/ohms/CBOFS". To build a new model system you will need a working hydrodynamic code which will be stored in /COMF/ohms/CBOFS/sorc. Its compiled executables will be located in /COMF/ohms/CBOFS/bin. All fixed files, such as grids, climatology files and control file templates will be stored in /COMF/ohms/CBOFS/info. A directory where the program will execute and drop runtime files must be created, /COMF/ohms/CBOFS/work. Also create the directories of execlog and docs. The archive directory will hold large amounts of output files and so is usually a logical link to another disk system:

*ln -s /archive/COMF/CBOFS/archive/ /COMF/ohms/CBOFS/archive/*

With all of these directories and a working hydrodynamic model you should now be able to proceed to build the standardized operational shell script.

## Appendix A 3 MODELCRONRUN.SH: THE CONTROLLING CRONTAB SCRIPT

A single script will run the COMF operational model. It will be executed via crontab control which will fully determine the running times of the model. The example script, MODELCRONRUN.sh, is called with a crontab file like:

*SETE=/COMF/oqcs/setenvironmentvariables_dsofs1.sh*
*MODELDIR=/COMF/ohms/CBOFS*

*# CBOFSNOW.sh Nowcast and Forecast launches*
*10 0,6,12,18 * * * source $SETE ; $MODELDIR/scripts/MAIN_CBOFS.sh &> *
*$MODELDIR/execlog/logcbofsMAIN*

The first two lines set environment variables which tell the scripts which directories the rest of the system should use. The setenvironmentvariables.sh script, described below, sets the path to include the executables of the COMF system. The MODELDIR variable specifies the home directory of the model itself. It is also possible to include these two variables inside the MODELCRONRUN.sh scripts.

The last line tells crontab to execute the script every day on hours 0, 6, 12, 18 at 10 minutes past the hour. The 10 minutes past the hour is useful so that the NWLON real time data, which can have a six minute delay, will be available.

The standard out and standard error messages are redirected to the log file in the model's execlog directory. This log file is overwritten every time the model is run. Other logging functionality is available within the script.

**Appendix A 3.1 MODULE 0: Set Up Environment Variables for Directories**

The first executed line of the model system must be the call to a setenvironmentvariables.sh script. This can be done either inside the MODELCRONRUN.sh script, inside the crontab file (as above), or from the command line prompt if running interactively.

*source /COMF/oqcs/setenvironmentvariables_dsofs.sh*

This is one of only two places in all of the scripts where the root directory of the machine upon which the model is being run is specified. In the example, /COMF/oqcs/ indicates the fully specified location of the setenvironmentvariables.sh script. It should be the same for all models running on the same machine. The purpose of the setenvironmentvariables.sh script is to specify the paths and directory locations for ALL resources of the computer and COMF necessary to run a COMF model. If a resource is not named in that file, you should not use it.

Next a series of directory names are specified which describe the location of the model files. MODELDIR is the root of the model, and is the only other place where the root directory of the computer is specified. The other directories should lie inside it. So the only line in this section which should be changed for a new model is:

*export MODELDIR=/COMF/ohms/CBOFS*

*export MODELWORK = $MODELDIR/work*

Usually the default directory is set to MODELWORK by doing
*cd $MODELWORK*

**Appendix A 3. 2 MODULE 1: Computer System Tests**

This section performs computer system tests and verifies that running the model is possible at this time. It creates CORMS flags to describe the available disk space on the computer for the attached disk drives of the COMF system. In this case they are COMF and odaas1. Those names may need to be changed for different computer systems. Verify by examining the df dump of the disk drives on the system. OFS_CONTROL.sh (Appendix B 31) is a script which will prevent the model from running multiple copies of it self. This is needed in case of a system slowdown or other major malfunction.

Some house keeping is done here by removing a few scratch files left over from the previous runs. The assumed directory is MODELWORK, but good practice when deleting files is to specify them with full directory names and do not use the greatly feared "rm *".

**Appendix A 3. 3 MODULE 2: Create the start and stop times**

Three times are needed to specify a nowcast/forecast cycle: the starting time of a nowcast, the ending time of the nowcast, which is also the starting time of the forecast, and finally the ending time of the forecast.

The starting time of the nowcast must be based on the time of the HOTSTART file which will be used. The modeler will have to provide a mechanism to find that value from his HOTSTART file. The first action of this section is therefore to locate the previous HOTSTART file and copy it to MODELWORK. Then the time_hotstart is pulled off the file, in this case using the program "readinitspace.x"

```
cp $MODELINIT/HOTSTART.DAT $MODELWORK/.
cp $MODELINIT/wlcbbtHOTSTART.dat $MODELWORK/.
# Start with time read from the hotstart file
time_hotstart=`readinitspace.x << EOD
"HOTSTART.DAT"
EOD`
```

The ending time of the nowcast is usually "now" time with the minutes rounded off.
*time_now=`date -u +"%Y %m %d %H 0"`*

The ending time of the forecast is obtained by simply adding the duration of the forecast (e.g. 24 hours) to time_now:
*time_forecastend=`datemath $time_now + 0 0 0 24 0`*

In the CBOFS case it was found that giving a slightly squishy ending time for the data acquisition programs was needed to guarantee that the nowcast input files were actually long enough. Another variable with 30 minutes added is used:

*`time_nowcastend=`date -u +"%Y %m %d %H 30"`*

Alter this section as needed, but the time variables must have these names: time_hotstart, time_now, time_forecastend.

### Appendix A 3. 4 MODULE 3: Get data

This is the section where the rubber hits the road, the input data section which calls all the middle-ware data grabbing routines which are the guts of COMF. However it ends up appearing reassuringly simple. All sources of input data for the upcoming run are specified here by database name, station id, starting time and ending time.

For CBOFS the only inputs are the water level specified at the mouth of the bay and two winds specified inside the bay. The Chesapeake Bay Bridge Tunnel water level is grabbed from the NWLON database using WLQCF.sh (Appendix B 53) with:

```
WLQCF.sh 8638863 NWLON "$time_hotstart" "$time_nowcastend" 0.10 \
        wlcbbt.dat wlcbbtHOTSTART.dat > $MODELLOGDIR/WLQCF.log
```

The two winds are grabbed for Thomas Point from NDBC and CBBT from NWLON using WINDQCF.sh (Appendix B 49):

```
WINDQCF.sh "TPLM2" NDBC "$time_hotstart" "$time_nowcastend" 0.10 \
        windtplm.dat > $MODELLOGDIR/WINDQCF1.log
WINDQCF.sh 8638863 NWLON "$time_hotstart" "$time_nowcastend" 0.10 \
        windcbbt.dat > $MODELLOGDIR/WINDQCF2.log
```

All the COMF data grabbing routines automatically add a CORMS flag to the $CORMSLOG. The grabbers all put out a little too much standard I/O so it has been redirected to log files for later examination if things go badly.

### Appendix A 3. 5 MODULE 4: Reformat data

The result of Module 3 will be a few data files which are in the standardized COMF file formats, ASCII time series or NetCDF files. All hydrodynamic models will require their own peculiar input files to be built from these files. This is done in the Reformat Module. The modeler is given a free hand in this section to include scripts and programs of their own design. Detailed comment statements and lists of input and output files should be included to allow for subsequent model maintenance.

In the CBOFS example the water level file is altered to provide the outer boundary condition water level file by using an awk command to alter the phase (by 17 min) and magnitude (MLLW changed to MSL-.442, Tide enhanced by 1.134) of the CBBT water levels to that needed at the outer oceanic boundary, and change the Gregorian y, m, d, h, min date format to the year, yearday format required by MECCA:

```
awk '{ print $1 " " $2 " " $3" " $4  " " $5-17 " " ($8+($9-.442)*1.134) }'\
        wlcbbt.dat | greg2yday.x > gentide_now.out
```

The wind fields for the CBOFS hydrodynamic model, MECCA, are strange binary files which must be generated by a Fortran program from the two WINDQCF.sh time series. The manipulation of the CBBT and Thomas Point wind files is to prepare them for the genwind_2obsoqcs.x Fortran reformator written specifically for MECCA.

After each reformatting operation is performed a CORMS flag should be formed to indicate the relative success of the operation. The gentide_now CORMS flags formed here are either 100% complete or 0% complete depending upon only whether the output file exists.

**Appendix A 3. 6 MODULE 5: Run the hydrodynamic model**

All of the input data files should now be available to execute the core hydrodynamic model. However the control file which tells a model how to run, what times to use and dozens of other runtime parameters still needs to be built. This section will build such a control file, execute the model core, and rename and move output files as needed.

For MECCA a now.con file is constructed by changing the year, month, day, hour and duration of the run in a template file. A sed command does this work. The fixed grid file is copied to the MODELWORK directory. Finally the mecca21nclf95.x binary executable is executed. Notice that it is in the PATH because PATH was augmented with $MODELBIN in Module 0.

After execution a test is performed to determine if the model ended correctly. For MECCA this involves checking for the phrase " ISTOP= 0" on the last line of the MECCA standard output file. A CORMS flag is generated, but the script is not terminated upon failure.

Resultant files are moved and renamed. This includes the HOTSTART file to be used for the next Forecast and for the next cycle's nowcast. A tar file of the input files is created for later archiving as well, modelinput.tar.

The main COMF output files are the NetCDF files, stationsnow.nc and fieldsnow.nc. These are standard names required by the ARCHIVE.sh script (Append B 1) which will be executed later. Specific model dependent names are removed in order to create these standards.

The NetCDF files should be created internally by the model code using the HYDRONetCDF subroutines (Appendix C 12-14). However if the model does not use the internal routines then a post-processing step should be added to the end of this module to create these files. No other output

format will be tolerated for COMF models and this must be provided to post any graphics or archive results.

### Appendix A 3. 7 FORECAST: Forecast Cycle Repeats 2-5

So far all of the operations of modules 2-5 have applied only to the nowcast. The Forecast will repeat all of these modules but will access forecast files and rename outputs appropriately.

Forecast Module 2 Set Times for Forecast

The timing module may or may not need to be present in forecast mode. For CBOFS the time_nowcastend is read from the nowcast hotstart file. This should not be necessary, but if the model ends on a non-hourly time step this could be a necessary precaution. The parameter time_forecastend is constructed by adding 24 hours.

Forecast Module 3 Grab Forecast Input Data

Extra Tropical Storm Surge model point guidance is grabbed for forecast water levels and NAM model wind forecast guidance are grabbed for forecast winds.

*WLQCF.sh 8638863 ETSS "$time_nowcastend" "$time_forecastend" 0.10 *
*wlcbbtfore.dat wlcbbtnow.dat > $MODELLOGDIR/WLQCF.log*

*WINDQCF.sh "-78 -74 36 40" NAM "$time_nowcastend" "$time_forecastend" 1.0 *
*windseta.nc > windetaqcf.log*

Forecast Module 4 Reformat Inputs

The water level is reformatted exactly as for the nowcast. The wind files for MECCA must be generated using a different version of "genwind" which reads the NetCDF file output.

Forecast Module 5 Run Model

Nearly identical run of the model as the nowcast, but with a different control file template. A CORMS flag for model successful run is generated. The output NetCDF files are moved to stationsfore.nc and fieldsfore.nc. The forecast input files are concatenated to modelinput.tar using the "tar -rvf" command. The HOTSTART from the forecast run can be discarded.

### Appendix A 3. 8 MODULE 6: Archive the data

The archive operation is automated by the script ARCHIVE.sh. This script expects to find files in the local directory ($MODELWORK) which end in standard name extensions. Input to ARCHIVE.sh is the date to use, which is usually the time at the end of the nowcast and the

beginning of the forecast, and the name of the model, or Operational Forecast System (OFS), in this case CBOFS.

*ARCHIVE.sh CBOFS "$time_nowcastend" "$time_nwocastend"*

For the CBOFS it is expecting to find and archive the files: hotstartout.dat, stationsnow.nc, fieldsnow.nc, stationsfore.nc, fieldsfore.nc and modelinput.tar. It will rename these with a date and hour and place them into a directory structure with date and hour conventions under the directory $ARCHIVEDIR, as specified in Module 0. From the ARCHIVE.sh header comments:

*$ARCHIVEDIR/modelinput/YYYYMM/YYYYMMDDHHMI_$OFS_NAME_modelinput*.tar*
*$ARCHIVEDIR/hotstart/YYYYMM/YYYYMMDDHHMI_$OFS_NAME_hotstartout*
*$ARCHIVEDIR/netcdf/YYYYMM/YYYYMMDDHHMI_$OFS_NAME_fields*.nc*
*$ARCHIVEDIR/netcdf/YYYYMM/YYYYMMDDHHMI_$OFS_NAME_stations*.nc*

The only change to make to this module will be the OFS name ($OFS_NAME) in the ARCHIVE.sh call.

**Appendix A 3. 9 MODULE 7: Create the graphics**

The standardized graphics system for producing the CO-OPS web page graphics is controlled by the GRAPHICS.sh script (Appendix B 17). It is a very complicated script, which has its own description section GRAPHICS.sh. The script reads control files for lists of stations and data to be plotted. It then automatically accesses the NWLON database for the observation data. It expects model result data in the NetCDF file formats. It is capable of concatenating model result NetCDF files to span longer time periods for the nowcast, if, for instance, the model runs a six hour nowcast four times a day, but the plotting will include 24 hours of nowcast.

The modeler will be responsible for creating the control files which specify the details of what graphics will be constructed. The control files will be created once and stored in $MODELINFO. The GRAPHICS.sh script expects to find control files in the local directory ($MODELWORK) called plot_time series_wl.ctl, plot_field.ctl and plot_curr.ctl. Another file, $MODELINFO/stationdata.dat, guides the download of observation data which will be plotted to compare with the model results. The plot*.ctl control files list dozens of parameters which guide the choices of plotting, such as labels, units and size of plots.

The running of the GRAPHICS.sh script requires some preparation in MODELCRONRUN.sh. The first step is to copy the control files from $MODELINFO to $MODELWORK. The program will also expect to find the file $MODELINFO/stationdata.dat. Finally GRAPHICS.sh expects the starting and ending times for the time series plots. These can be different from the run times of the model, especially if the model is run every six hours, but we desire plots which span the previous 24 hours.

The section ends with the construction of a CORMS flag and call of ARCHIVE_GRAPHICS.sh (Appendix B 2) which will archive the many graphics files and copy them to the web page

directory $MODELWWW (as specified in Module 0). As soon as they are copied to the $MODELWWW they become part of the PORTS web page.

To summarize, the modeler must create and edit the control files and stationdata.dat. Then the modeler will change the Module 7 section by copying the control files to be used into $MODELWORK and specifying the start and stop times of the graphics:

```
cp $MODELINFO/plot_timeseries_cbofs.ctl plot_timeseries_wl.ctl
cp $MODELINFO/plot_field_cbofs2.ctl plot_field.ctl
TIME_NOWCASTSTART=`datemath $time_roundhour - 0 0 0 24 0 `
GRAPHICS.sh  "$TIME_NOWCASTSTART" "$time_forecastend"  >  \
        $MODELLOGDIR/graphics.log
```

### Appendix A 3. 10 MODULE 8: Create the CORMS FLAGS

The modeler has nothing to do in this section of MODELCRONRUN.sh. It calls MAKECORMSFLAGS.sh (Appendix B 23) which finalizes the CORMS flags and copies them to the CORMS web page handler. The modeler is responsible for the CORMS flag control file, $MODELINFO/corms_table.txt. This file has the name of all the CORMS flags to be generated and passed to CORMS real-time monitoring computer system. It specifies the percentage values which create green, yellow or red CORMS flags.

```
1 DISKFREE NGOFS 15 30
2 DISKFREE ODAAS 15 30
3 WL 8531680     60 80
4 WL 8516945     60 80
5 WIND 8531680    30 80
...
```

This specifies that the WL 8531680 flag will be red if less than 60% of the expected data is received will be yellow if between 60% and 80% and green if more than 80% was received.

### Appendix A 3. 11 MODULE 9: Purge old files

The PURGE.sh control file, PURGE.ctl, needs to be set to conserve disk space. This file is located under the directory $MODELINFO. An example is the CBOFS PURGE.ctl:

```
# CBOFS purge control file
# Each line has 3 fields: first is the archive subdirectory corresponding to file type,
# second is a string representing the part of the file name after the date string (can have wildcards),
# last is an integer for how many days old the files must be before being purged
execlog *diagnostics*log 10
netcdf *CBOFS_fields_*.nc 30
cormsflags *corms*txt 90
graphics *CBOFS_*.png 30
```

The lines with # at the beginning are comments. The fifth line sets files in the execlog directory with names that include a date prefix, "diagnostics" in the middle of the file name, and the suffix "log", to be removed after 10 days.

## Appendix A 4 TESTING A NEW SCRIPT

Testing a new script usually proceeds in three stages: testing individual script calls, just making sure it runs at all, and testing as scheduled job to discover errors which might occur during daily, routine operations.

To test individual lines one should copy and paste Module 0 to the command line. This enables your command session to behave like the inside of the script. The most significant effect was the sourcing of setenvironmentvariables.sh which put all of the COMF system into your path. Without doing that most scripts will not even be located, much less behave correctly. Then just copy and paste the time variables. Complicated scripts like WLQCF.sh can be tested one line at a time. Quite a bit can be learned this way, but eventually you need to run the full system. (The developer should be using either the bash or sh UNIX shells.)

To test the whole system at once the script can be run from the command line, as it is a standalone script. However it will produce a few files which will interfere with doing another test a few minutes later after a small change has been made. To prevent this sort of problem you probably need to suppress the overwriting of the old nowcast Hotstart file. This can be accomplished by executing the lines in Module 2:

*cp $MODELINIT/HOTSTART.DAT $MODELWORK/.*
*cp $MODELINIT/wlcbbtHOTSTART.dat $MODELWORK/.*

by hand at the command line once, to move the HOTSTART files into the $MODELWORK. Then comment them out of the MODELCRONRUN.sh script so that they do not get overwritten.

It might also be advisable to redirect output of the test cases. In Module 0 the output directories can be assigned to something different, like:

*export ARCHIVEDIR=$MODELDIR/archive_test*
*export MODELWWW=$MODELDIR/wwwgraphics_test*

Outputs to the PORTS pages should be suppressed by commenting the call to ARCHIVE_GRAPHICS.sh. It can also annoy the CORMS people to see lots of random messages flying across their screens, so I usually suppress the Module 8 call to MAKECORMSFLAGS.sh.

For burn in testing the crontab script should be made fully functional except, perhaps, for the ARCHIVE_GRAPHICS.sh.and MAKECORMSFLAGS.sh.

These types of tests should be run in a CVS'ed directory structure. See CVS and Testing (Chapter 7 of main report). To create a new CVS repository the modeler should build as much as the directory structure and files as possible. Then delete all the scratch files and backups of old testing programs to clean up the directories. The creation (import) of a CVS repository is then accomplished with these commands:

*Export CVSROOT=:ext:modellers_login_name@dsofs1.nos-tcn.noaa.gov:/comf/CVSPROJECTS/*
*export CVS_RSH=ssh*
*echo $CVSROOT*
*cd ohms/NEWmodel*
*cvs import -m "NEW model from modellers_login_name" CBOFS NEWmodel CSDLMMAP start*

## Appendix A 5 MOVING THE SYSTEM BETWEEN COMPUTERS

After scheduled testing on a test computer has been completed the model can be lifted and moved to a new machine rather easily. The only differences should appear in Module 0 where directories are named. If the CVS system was used in testing then the new copy of the system may be place in a staging location using the CVS checkout command:

*cd /comf/staging/COMF/ohms*
*cvs co CBOFS*

Rebuild all the missing directories as indicated in step 2 above. The link for the archive and wwwachive directories may be made to a convenient location.

Edit Module 0 for the script to point at the new directory system. That should only be for setenvironmentvariables.sh and MODELDIR. The Module 1 computer system tests for the new computer's disk drives may also need to be changed.

Finally create the crontab command to start it running. The file CRON_cbofs.sh (the model version of MODELCRONRUN.sh) contains the crontab commands.

*crontab -r*
*crontab $MODELDIR/scripts/CRON_cbofs.sh*

## Appendix A 6 CONCLUSIONS

The purpose of these standards is to build multiple models which can be maintained as a group without needing to know about the interior of each model. By following these guidelines we think this goal can be achieved.

Future enhancements to the system will be a more extensive development environment where changes to the scripts will be tested and carefully transferred to the operational systems. The basis of this will be a version control system and mirrored models running on separate development

computers. Again, this is possible only through the rigorous use of the relocate-able directories specified via global environment variables. Another future enhancement will be a runtime control system to replace the crontab method. However the simplicity achieved by our use of the MODELCRONRUN.sh script should allow the implementation of such a method quite easily.

## APPENDIX B. SCRIPTS LIBRARY

### TABLE OF CONTENTS


Appendix B 1 Program Name: ARCHIVE.sh ..... 53
Appendix B 2 Program Name: ARCHIVE_GRAPHICS.sh ..... 54
Appendix B 3 Script Name: AT_read_nwlonweb.sh ..... 55
Appendix B 4 Script Name: concatlist.sh ..... 56
Appendix B 5 Script Name: concatnetcdf.sh ..... 57
Appendix B 6 Script Name: cormspercent.sh ..... 58
Appendix B 7 Script Name: CORMSPROCESS.pl ..... 59
Appendix B 8 Script Name: CRON_NAM_nc.sh ..... 60
Appendix B 9 Script Name: CURRQCF.sh ..... 61
Appendix B 10 Script Name: get_data_npdb_currents.sh ..... 62
Appendix B 11 Script Name: get_data_nwlon_db.sh ..... 63
Appendix B 12 Program Name: get_glsea.pl ..... 64
Appendix B 13 Script Name: get_harmonics.sh ..... 65
Appendix B 14 Program Name: get_sfcmarobs.pl ..... 66
Appendix B 15 Script Name: grabarchivenetcdf.sh ..... 67
Appendix B 16 Script Name: grabarchivenetcdf_fore.sh ..... 68
Appendix B 17 Script Name: GRAPHICS.sh ..... 69
Appendix B 18 Control file: plot_field.ctl ..... 70
Appendix B 19 Control file: plot_timeseries_wl.ctl ..... 72
Appendix B 20 Script Name: hotstart_copy.sh ..... 74
Appendix B 21 Script Name: how_new.sh ..... 75
Appendix B 22 Script Name: how_old.sh ..... 76
Appendix B 23 Script Name: MAKECORMSFLAGS.sh ..... 77
Appendix B 24 Script Name: MODELCRONRUN.sh ..... 78
Appendix B 25 Script Name: NCLwindgetNAMstation.sh ..... 79
Appendix B 26 Script Name: NCLwindgetNAMsub.sh ..... 80
Appendix B 27 Script Name: NetCDFgetstation_currents.sh ..... 81
Appendix B 28 Script Name: NetCDFgetstation_nwlon_fast.sh ..... 82
Appendix B 29 Script Name: NetCDFgetstations_astro.sh ..... 83
Appendix B 30 Script Name: notbracket.pl ..... 84
Appendix B 31 Program Name: OFS_CONTROL.sh ..... 85
Appendix B 32 Script Name: pres_read_nwlonweb.sh ..... 86
Appendix B 33 Script Name: PRESQCF.sh ..... 87
Appendix B 34 Program Name: PURGE.sh ..... 88
Appendix B 35 Script Name: read_ndbc_archive.sh ..... 89
Appendix B 36 Script Name: READUSGS.pl ..... 90
Appendix B 37 Script Name: river_read_usgs.sh ..... 91
Appendix B 38 Script Name: river_read_usgsarchive.sh ..... 92
Appendix B 39 Script Name: river_read_usgsmysql.sh ..... 93
Appendix B 40 Script Name: RIVERQCF.sh ..... 94
Appendix B 41 Script Name: setenvironmentvariables.sh ..... 95
Appendix B 42 Script Name: SALINITY.pl ..... 96

Appendix B 43 Script Name: SALTQCF.sh ..... 97
Appendix B 44 Script Name: temp_read_ndbc.sh ..... 98
Appendix B 45 Script Name: TEMPQCF.sh ..... 99
Appendix B 46 Script Name: tide_read_nwlonweb.sh ..... 100
Appendix B 47 Script Name: wind_read_ndbc.sh ..... 101
Appendix B 48 Script Name: wind_read_nwlonweb.sh ..... 102
Appendix B 49 Script Name: WINDQCF.sh ..... 103
Appendix B 50 Script Name: wl_read_etss.sh ..... 104
Appendix B 51 Script Name: wl_read_nwlonweb.sh ..... 105
Appendix B 52 Script Name: wl_read_nwlonwebv.sh ..... 106
Appendix B 53 Script Name: WLQCF.sh ..... 107
Appendix B 54 Script Name: WT_read_nwlonweb.sh ..... 108

**Appendix B 1 Program Name: ARCHIVE.sh**

Location: /COMF/oqcs/scripts/

Technical Contact: Zack Bronder — Org: NOS/CO-OPS
Phone: 301-713-2890 x152 — E-Mail: Zachary.Bronder@noaa.gov
Mark Vincent — Org: NOS/CO-OPS
Phone: 301-713-2890 X151 — E-Mail: Mark.Vincent@noaa.gov

Abstract:

A standard part of COMF, ARCHIVE.sh is used to archive output from NOS operational coastal forecast systems.

Function: Stores files found in local directory:
modelinput*.tar hotstartout* graphics*.tar
fields*.nc stations*.nc fields*.grb stations*.grb
to the directory $ARCHIVEDIR under subdirectories and names:

$ARCHIVEDIR/modelinput/YYYYMM/YYYYMMDDHHMI_$OFS_NAME_modelinput*.tar
$ARCHIVEDIR/hotstart/YYYYMM/YYYYMMDDHHMI_$OFS_NAME_hotstartout
$ARCHIVEDIR/graphics/YYYYMM/YYYYMMDDHHMI_$OFS_NAME_graphics*.tar
$ARCHIVEDIR/netcdf/YYYYMM/YYYYMMDDHHMI_$OFS_NAME_fields*.nc
$ARCHIVEDIR/netcdf/YYYYMM/YYYYMMDDHHMI_$OFS_NAME_stations*.nc
$ARCHIVEDIR/grib/YYYYMM/YYYYMMDDHHMI_$OFS_NAME_fields*.grb
$ARCHIVEDIR/grib/YYYYMM/YYYYMMDDHHMI_$OFS_NAME_stations*.grb

Usage:

Interactively: ARCHIVE.sh $OFS_NAME $TIME_NOWCASTEND $TIME_HOTSTART
ARCHIVE.sh 'TBOFS' '2000 01 31 23 59' '2000 01 31 23 00'

Automatically: ARCHIVE.sh will be called by MAIN_??OFS.sh and run as a cron job.

Input Parameters: YYYY=year, MM=month (ex. 03), DD=day of month (ex. 05),
HH=hour (UTC) (ex. 06), MI=minute (ex 00-59)

Language: Bourne Shell Script

Programs Called:

| Name | Location | Description |
|---|---|---|
| dateformat | /ngofs/oqcs/binlinux/dateformat | A C program that formats date strings. |

Input Files:

| Name | Location | Description |
|---|---|---|
| *graphics*.tar | $MODELDIR/work/ | Tarred Model graphics files |
| hotstart* | $MODELDIR/work/ | Model hotstart file |
| *.nc | $MODELDIR/work/ | Model NetCDF files |
| modelinput*.tar | $MODELDIR/work/ | Tarred model input files |
| grb | $MODELDIR/work/ | Grib files |

Output Files:

| Name | Location | Description |
|---|---|---|
| YYYYMMDDHHMI_graphics.tar | $ARCHIVEDIR/graphics/YYYYMM/ | Archived graphics |
| YYYYMMDDHHMI_hotstart | $ARCHIVEDIR/hotstart/YYYYMM/ | Archived hotstart |
| YYYYMMDDHHMI_*.nc | $ARCHIVEDIR/netcdf/YYYYMM/ | Archived netCDF's |
| YYYYMMDDHHMI_*input*.tar | $ARCHIVEDIR/input/YYYYMM/ | Archived inputs |
| YYYYMMDDHHMI_*.grb | $ARCHIVEDIR/grib/YYYYMM/ | Grib files |

Author: Zack Bronder — Date: August 1, 2003

**Appendix B 2 Program Name: ARCHIVE_GRAPHICS.sh**

Location: /COMF/oqcs/scripts/

Technical Contact: Zack Bronder — Org: NOS/CO-OPS
Phone: 301-713-2890 x152 — E-Mail: Zachary.Bronder@noaa.gov
Mark Vincent — Org: NOS/CO-OPS
Phone: 301-713-2890 X151 — E-Mail: Mark.Vincent@noaa.gov

Abstract:

A standard part of COMF, ARCHIVE_GRAPHICS.sh is used to archive graphics from NOS operational coastal forecast systems.

Usage: Interactively: ARCHIVE_GRAPHICS.sh $OFS_NAME $TIME_NOWCASTEND
ARCHIVE_GRAPHICS.sh '2000 01 31 23 59'
Automatically: Called by MAIN_??OFS.sh and run as a cron job.

Input Parameters: YYYY=year, MM=month (ex. 03), DD=day of month (ex. 05),
HH=hour (UTC) (ex. 06), MI=minute (ex 00-59)

Language: Bourne Shell Script

Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Programs Called:

| Name | Location | Description |
|---|---|---|
| dateformat | /ngofs/oqcs/binlinux/dateformat | A C program that formats date strings. |

Input Files:

| Name | Location | Description |
|---|---|---|
| *.png | $MODELDIR/work/ | Model graphics files |

Output Files:

| Name | Location |
|---|---|
| YYYYMMDDHHMI_$ofsname*.png | $ARCHIVEDIR/graphics/YYYYMM/DDHHMI/ |

Libraries Used: None

Author Name: Zack Bronder Creation Date: February 25, 2004

Remarks: First Draft of this script. Some file names may change. Maybe grib files to archive.
Use environment variable as date inputs. There may be changes in naming convention.
This script needs the following environment variables to be defined:
$ARCHIVEDIR

**Appendix B 3 Script Name: AT_read_nwlonweb.sh**

Directory Location: /COMF/oqcs/scripts

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Aijun Zhang Org: NOS/CSDL
Phone: 301-713-2809x113 E-Mail: aijun.zhang@noaa.gov

Abstract:

Grab the air temperature data from the NWLON web site. Retrieve Meteorological Oceanographic Data web page. This uses a screen scraper which directly calls the CGI used to fill in the data from
http://co-ops.nos.noaa.gov/data_retrieve.shtml?input_code=101000111pan
This is an web version and backup to get_data_nwlon_db.sh. It depends upon this line:
echo "http://www.co-ops.nos.noaa.gov/cgi-bin/co-ops_qry_direct.cgi?\
stn=$stnid&dcp=1&ssid=D1&pc=W1&datum=NULL&unit=0&bdate=$bdate\
&edate=$edate&date=3&shift=0&level=1&form=0&host=&addr=10.60.5.243\
&data_type=pan&format=View+Data" > $REQUESTGET
As with all screen scrapers if CO-OPS changes this reference, then this program will crash.
Output file has date, forecasthour, air temperature

(nwlon, tide forecasthour ==0 )

```
y   m d h m   fh   at
2002 12 30 12 30   0   3.7000
```

Usage: Interactively: AT_read_nwlonweb.sh stationid startdate enddate outputfilename
Via cron: Called by TEMPQCF.sh.

Input Parameters:
station id Ex. 8638610
starting date Ex. "2002 12 10 00 00"
ending date Ex. "2002 12 12 12 00"
output file name Ex. at8638610.txt

Language: Bourne Shell Script
Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| dateformat | /COMF/oqcs/bin...sorc | Flexible String builder using dates. |
| mktemp | /COMF/oqcs/bin../sorc | Make a temporary unique filename. |
| wget | /COMF/oqcs/binsgi | Web Grabber. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| $4 | Depend on requests. | Output file with date, forecasthour, air temperature |

Author Name: Aijun Zhang Creation Date: 2005-01-25

**Appendix B 4 Script Name: concatlist.sh**

Directory Location: /COMF/oqcs/scripts/

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Abstract:
Script of reading a list of NetCDF station files to be concatenated.

Usage: interactively: concatlist.sh "$LIST" $OUTPUT
via cron: called by grabarchivenetcdf.sh

Input Parameters: $LIST=`ls station*nc`, for example,
$OUTPUT is the output NetCDF filename
IMPORTANT: remember to put the quotes around $LIST

Language: Bourne Shell Script

Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Author Name: Tom Gross Creation Date: 2003

**Appendix B 5 Script Name: concatnetcdf.sh**

Directory Location: /COMF/oqcs/scripts/

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Abstract:
Script for reading a list of NetCDF station files to be concatenated.

Usage: concatnetcdf.sh "$LIST" $OUTPUT

Input Parameters: $LIST=`ls station*nc`, for example,
$OUTPUT is the output NetCDF filename
IMPORTANT: remember to put the quotes around $LIST

Language: Bourne Shell Script

Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Author Name: Tom Gross Creation Date: 2003

**Appendix B 6 Script Name: cormspercent.sh**

Directory Location: /COMF/oqcs/scripts/

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Abstract: Calculates the percentage of data contained inside the data file returned directly from the database. This is called by the *QCF.sh scripts just after grabbing the raw data and before it is gap filled and resampled to DT. The output is used for the CORMS flag generation.
It works by simply counting the lines in the file (between the dates).
It calculates how many lines should be expected by doing some datemath with tstart, tend and DT .

Usage: Interactively: CORMSPERCENT=`cormspercent.sh "$tstart" "$tend" $DT $filename`
echo "WL "$sid" "$CORMSPERCENT >> $CORMSLOG
Via cron: Called by the *QCF.sh, like WLQCF.sh

Input Parameters: $1 : $tstart, start time, "2005 01 24 12 00"
$2 : $tend, end time, "2005 01 31 12 00"
$3 : $DT, time interval, 1 = 1 hour, 0.1 = 6 minutes.
$4 : $filename, file name of which contains comrs percent results.

Language: Bourne Shell Script

Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| mktemp | /COMF/oqcs/binsgi | Makes a temporary unique filename. |
| datemath | /COMF/oqcs/bin.../sorc | Do simple addition, subtraction of dates. |
| dateformat | /COMF/oqcs/bin.../sor | Flexible String builder using dates. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| cormslogfile.txt | /COMF/info/exelog | Text log message file. |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 7 Script Name: CORMSPROCESS.pl**

Directory Location: /COMF/oqcs/scripts/

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Abstract:

Process the $CORMSLOG file into a simple string of GYR for the Green, Yellow and Red CORMS buttons to be displayed.

Processes a file of corms percentages found in the input file cormsfile. Use the information in cormstable to point to flags and percentages. Outputs just the flag values to cormsflags The forecast files are judged by their age. Negative hours means the forecast is too old by that many hours so the yellow, red cutoffs might be -24 -12 NAM winds are recieved only every six hours and they are already four hours old when you get them, so quite often the flag will be in the range -10 : -4 and that would be good!

Usage:

Interactively: CORMSPROCESS.pl $MODELDIR/INFO/cormstable $CORMSLOG(cormsfile) \
cormsflags "$time_nowcastend"

Via cron: Called by MAKECORMSFLAGS.sh

Input Parameters:

$1 : $MODELINFO/cormstable: has multiple lines like:

```
1 WL 8638863      60 80
2 WIND 8638863    60 80
3 CURR 8638863    60 80
4 SALT 8638863    60 80
5 TEMP 8638863    60 80
```

These indicate the i'th flag is red<60 <yellow< 80 < green
Gray for -999.99. Black for all other cases.
Higher percentages are always better.

$2: $CORMSLOG: is a file of corms percentages. has multiple lines like:

```
WL 8638863 97.0954
WIND 8638863 70.0954
CURR 8638863 50.0954
SALT 8638863 40.0954
TEMP 8638863 -999.99
```

$3 : cormsflags: gray, red, yellow, green, black

$4 : time_nowcastend : "2005 01 24 12 00"

Language: Perl Script

Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Input Files:

| Name | Directory Location | Description |
|---|---|---|
| corms_table.txt | $MODELDIR/info/ | A text corm flags table file. |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 8 Script Name: CRON_NAM_nc.sh**

Directory Location: /COMF/oqcs/scripts

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Abstract:

Runs the grib to NetCDF translator four times a day.
Produces the NetCDF wind file in /COMF/oqcs/archive/NAMnc.

Usage: Interactively: CRON_NAM_nc.sh
Via cron: N/A

Input Parameters: none

Language: Bourne Shell Script

Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| NCLetagrib2netcdf.sh | /COMF/oqcs/scripts/ | Transfers Grib file to NetCDF file. It is useless according to ODAAS. |
| CBOFSINITohms.sh | /COMF/ohms/cbofs/scripts/ | Initial model. |
| CBOFSNOWohms.sh | /COMF/ohms/cbofs/scripts/ | cold start script. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| NCLetagrib2netcdf.log | /COMF/oqcs/execlog/ | Message log file. |
| logcbofsINIT | /COMF/ohms/cbofs/execlog/ | Model log file. |
| logcbofsNOW | /COMF/ohms/cbofs/execlog/ | Model log file. |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 9 Script Name: CURRQCF.sh**

Directory Location: /COMF/oqcs/scripts

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Abstract:

Returns surface water currents m/s.
Standard T2 file:
y m d h min fh Ueastward, Vnorthward (m/s)
2002 12 29 12 30 0 0.53 -0.26
Data base:
NPDB
Return a QC'd, gap filled time series of data
Does current data
Parse out option to select different data base
Raw data read from ODAAS using particular data base

Usage: Interactively: CURRQCF.sh stationid database tstart tend DT binnum CURFILE
Via cron: Called by MODELCRONRUN.sh

Input Parameters: $1: station id (g01010)
$2: database name (NWLON)
$3: start time (YYYY MM DD hh mm)
$4: end time (YYYY MM DD hh mm)
$5: time interval (0.1)
$6: bin number
$7: output data filename(output.dat)

Language: Bourne Shell Script
Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| get_data_nwlon_db.sh | /COMF/oqcs/scripts | Reads NWLON data using SYBASE interface. |
| cormspercent.sh | /COMF/oqcs/scripts | Calculates percentage for CORMS flags. |
| datemath | /COMF/oqcs/sorc | Do simple addition, subtraction of dates. |
| mktemp.c | /COMF/oqcs/sorc | Makes a temporary unique filename. |
| wind_QC_station_gapfill.f | /COMF/oqcs/sorc | Wind edit and gap filler. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| $7 | User defined. | Standard T2 file: |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 10 Script Name: get_data_npdb_currents.sh**

Directory Location: /COMF/oqcs/scripts

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Abstract: Used to access the National PORTS Database to get current data

Usage:
Interactively:
get_data_npdb_currents.sh g01010 "2003 02 05 00 00" "2003 02 15 01 00" 5 outputfile
get_data_npdb_currents.sh g02010 "2005 01 23 00 00" "2005 01 24 12 00" 3 curr.out
Via cron: Called by CURRQCF.sh
Called by NetCDFgetstation_currents.sh

Input Parameters: $1: station id (g01010)
$2: start time (YYYY MM DD hh mm)
$3: end time (YYYY MM DD hh mm)
$4: bin number
$5: output data filename (output.dat)

Language: Bourne Shell Script

Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| isql | /opt/sybase-12.5/OCS/bin/ | Access the SYBASE database. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| $5 | User defined. | Standard TS2 file: |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 11 Script Name: get_data_nwlon_db.sh**

Directory Location: /COMF/oqcs/scripts

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Abstract: get_data_nwlon_db.sh is an all purpose shell script used to access the NWLON database written by Zhong to give us access to the Sybase NWLON database. It will become an all purpose reader grabbing different data by setting the data type flag. It forms an ISQL query command for the Sybase database. ISQL must be available on your machine and the environment

variable SYBASE set, i.e.:
export SYBASE=/opt/sybase-12.5
export PATH=$SYBASE/OCS/bin:$PATH

Choices for $4 data type:

| | | |
|---|---|---|
| WL | Water Level MLLW observations. | date, 1 float, 5 flags |
| WLPRED | Water Level MLLW Tidal Predictions. | date, 1 float, - flags |
| AP | Air Pressure | date, 1 float, 3 flags |
| WT | Water Temperature | date, 1 float, 3 flags |
| AT | Air Temperature | date, 1 float, 3 flags |
| WC | Surface Salinity | date, 1 float, 3 flags |
| WIND | Wind Observations Ueast, Vnorth | date, 2 float, 3 flags |

Two tmp files (tmpinput, tmpoutput) will be made in the directory where you presently run the script. Now the script only returns water level data with setting WL. The outputfile format is just exactly as required. MLLW water level data are provided with format f10.6 in unit meter. Time is UTC time. There are five flags for water level.

Flag1 -- set to 1: either the maximum or minimum water level height limit was exceeded.
Flag 2 -- when set to 1 indicates that the flat tolerance limit was exceeded
Flag 3 -- when set to 1 indicates that the rate of change tolerance limit was exceeded
Flag 4 -- set to 1 indicates that the temperature difference tolerance limit was exceeded
Flag 5 -- when set to 1 indicates that the height correction tolerance limit was exceeded

Usage:

Interactively: get_data_nwlon_db.sh $stationid "$tgrabstart" "$tgrabend" WL $SCRATCH
Via cron: Called by NetCDFgetstation_nwlon_fast.sh, PRESQCF.sh, SALTQCF.sh,
Called by TEMPQCF.sh, WINDQCF.sh, WLQCF.sh, wl_read_nwlonsybase.sh

Input Parameters: $1: station id : $2: start time (YYYY MM DD hh mm)
$3: end time (YYYY MM DD hh mm) ; $4: data type
$5: output data filename

Language: Bourne Shell Script

Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| mktemp | /COMF/oqcs/binsgi | Makes a temporary unique filename. |
| datemath | /COMF/oqcs/bin.../sorc | Do simple addition, subtraction of dates. |
| dateformat | /COMF/oqcs/bin.../sorc | Flexible String builder using dates. |
| tide_read_nwlonweb.sh | /COMF/oqcs/scripts | Grabs tide data from CO-OPS web site. |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 12 Program Name: get_glsea.pl**

Technical Contacts: Zack Bronder Org: NOAA/NOS/CO-OPS
Phone: 301-713-2890x152 E-mail: Zachary.Bronder@noaa.gov
Mark Vincent Org: NOAA/NOS/CO-OPS
Phone: 301-713-2890x151 E-mail: Mark.Vincent@noaa.gov

Directory Location: /COMF/oqcs/scripts

Abstract:

This script is used to read GLRL Great Lakes mean surface temperature. It outputs an ascii file. It is a part of COMF (formerly NGOFS), and it gets observation files from ODAAS.

Language: Perl script

Usage: Interactively: get_glsea.pl time lake nowcastend
get_glsea.pl '2005 01 01 00 00' ERIE '2005 01 02 00 00'
Automatically: get_glsea.pl can be called by model scripts,
such as MAIN_LEOFS.sh, which are launched via cron.

Input Parameters:

"time" is the date from which the mean surface temperature will be returned.
It consists of integers for year, month, day, hour, and minute.
In model scripts it will usually be $time_hotstart.
"lake" is the great lake at which the mean surface temperature will be returned.
"nowcastend" is the time of the end of nowcast. This is used to make CORMS flags.

Target computer: Runs on COMF computers, such as glofs.nos.noaa.gov.
Get input from ODAAS computers, such as odaas1.nos.noaa.gov.

Input Files:

| Name | Directory Location | Description |
|---|---|---|
| YYYYMMDD_glsea-tmps.txt | $ODAASDIR/ocean/obs/ncep/archives/YYYYMM | obs text file |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| get_glsea.txt | User defined. | Text file of lake surface temperature |

Author: Zack Bronder Creation Date: March 18, 2005

**Appendix B 13 Script Name: get_harmonics.sh**

Directory Location: COMF/oqcs/scripts

| | | |
|---|---|---|
| Technical Contact: | Zack Bronder | Org: NOS |
| | Phone: 301-713-2890x152 | Email: Zachary.Bronder@noaa.gov |
| | Greg Mott | Org: NOS |
| | Pone: | Email: Greg.Mott@noaa.gov |
| | Mark Vincent | Org: NOS |
| | Phone: 301-713-2890x151 | Email: Mark.Vincent@noaa.gov |

Abstract:

This script gets tidal constituents for a station ID that the user specifies, from the CO-OPS database.

Usage: Interactively: get_harmonics.sh <station_id>
Often times the ouput is redirected to
$OQCSDIR/info/predictions/lib/<station_id>.dat

Input Parameters: station_id

Language: Bourne Shell Script

Target Computer: Runs on COMF computers, such as glofs.nos.noaa.gov.

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| isql | /opt/sybase-12.5/OCS/bin/ | Access the SYBASE database. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| <station_id>.dat (optional) | $OQCSDIR/info/predictions/lib/ | data file. |

Author Name: Zack Bronder Creation Date: 2005-01-27

**Appendix B 14 Program Name: get_sfcmarobs.pl**

Abstract:

This script is used to concatenate all of NCEP surface marine observation ASCII text files from a time span specified by the user. It is a part of COMF (formerly NGOFS), and it gets obs files from ODAAS.

Location: $OQCSDIR/scripts/

Technical Contacts:

| | |
|---|---|
| Zack Bronder | Org: NOAA/NOS/CO-OPS |
| Phone: 301-713-2890x152 | E-mail: Zachary.Bronder@noaa.gov |
| Mark Vincent | Org: NOAA/NOS/CO-OPS |
| Phone: 301-713-2890x151 | E-mail: Mark.Vincent@noaa.gov |

Language: Perl

Usage: Interactively: get_sfcmarobs.pl start_time end_time
get_sfcmarobs.pl '2004 12 31 18 00' '2005 01 01 06 00'
Automatically: get_sfcmarobs.pl can be called by model scripts,
such as MAIN_LEOFS.sh, which are launched via cron.

Input Parameters: start_time is the start of the time span of the input obs files.
It consists of integers for year, month, day, hour, and minute.
In model scripts it will usually be $time_hotstart.
end_time is the end of the time span and has the same format.
In model scripts it will usually be $time_nowcastend.

Target computer: Runs on COMF computers, such as glofs.nos.noaa.gov.
Get input from ODAAS computers, such as odaas1.nos.noaa.gov.

Input Files:

| Name | Directory Location | Description |
|---|---|---|
| YYYMMDDHHsfcmarobs.txt | $ODAASDIR/atmos/obs/ncep/archives/YYYYMM | obs text file |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| get_sfcmarobs.txt | User defined | text file of several concatenated input files |
| get_sfcmarobs.lst | User defined | list of input files that are concatenated |

Author: Zack Bronder Creation Date: November 29, 2004

**Appendix B 15 Script Name: grabarchivenetcdf.sh**

Directory Location: /COMF/oqcs/scripts
Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Abstract: script for grabbing all nowcast.nc between a date range
Function:
Loops by hour through all file names between "$tstart" "$tend"
$ARCHIVEDIR/netcdf/%Y%m/%Y%m0%H$filetail
It identifies which files exist and puts the names in a list
The list is handed to concatnetcdf.sh,
which concatenates the NetCDF files into a single NetCDF output

Usage:
Interactively: grabarchivenetcdf.sh "$tstart" "$tend" $filetail $outputfilename
$outputfilename must end in .nc It will be a NetCDFfile
grabarchivenetcdf.sh "2003 11 07 12 0" "2003 11 08 12 00" _CBOFS2_stationsnow.nc cbofs.nc
This will grab all files with names like:
$ARCHIVEDIR/netcdf/200311/200311071200_CBOFS2_stationsnow.nc
Via cron: Called by MAIN_MODEL.sh

Input Parameters: starting time Ex. "2003 11 07 12 0"
Ending time Ex. "2003 11 08 12 00"
file name tail Ex. _CBOFS2_stationsnow.nc
output file name Ex. cbofs.nc

Language: Bourne Shell Script
Target Computer: Runs on COMF computers, such as dsofs1.nos.noaa.gov.

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| datemath | /COMF/oqcs/bin.../sorc | Do simple addition, subtraction of dates. |
| dateformat | /COMF/oqcs/bin.../sorc | Flexible String builder using dates. |
| concatlist.sh | /COMF/oqcs/binlinux | Reads a list of NetCDFfiles to be concatenated. |

Author Name: Tom Gross Creation Date: 2003

Remarks:
BONUS FEATURE:
The gbofs nowcast files have an hour stuck in their extension string
200311131400_GBOFS_stations_hsc.NOW.14.nc
This can be handled with dateformat syntax for the hour %H :
grabarchivenetcdf.sh "$tstart" "$tend" _GBOFS_stations_hsc.NOW.%H.nc gbofs.nc

It makes you think that this routine could be generalized by specifying the full path with that sort of syntax.

**Appendix B 16 Script Name: grabarchivenetcdf_fore.sh**

Directory Location: /COMF/oqcs/scripts
Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Abstract:

Script for grabbing the most recent forecast files. Find the one just before the $tnow date given.
Function:
Loops by hour through all file names between "$tnow"
and up to 96 hours previously
$ARCHIVEDIR/netcdf/%Y%m/%Y%m0%H$filetail

Usage: Interactively: grabarchivenetcdf_fore.sh "$tnow" $filetail $outputfilename
grabarchivenetcdf_fore.sh "2003 11 08 12 00" _CBOFS2_stationsfore.nc cbofsfore.nc
Via cron: Called by MAIN_MODEL.sh

Input Parameters: time now Ex. "2003 11 07 12 00"
file name tail Ex. _CBOFS2_stationsnow.nc
output file name Ex. cbofsfore.nc

Language: Bourne Shell Script
Target Computer: Runs on COMF computers, such as dsofs1.nos.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| datemath | /COMF/oqcs/bin.../sorc | Do simple addition, subtraction of dates. |
| dateformat | /COMF/oqcs/bin.../sorc | Flexible String builder using dates. |

Author Name: Tom Gross Creation Date: 2003

Remarks:

BONUS FEATURE:

The gbofs nowcast files have an hour stuck in their extension string
200311131400_GBOFS_stations_hsc.NOW.14.nc
This can be handled with dateformat syntax for the hour %H :
grabarchivenetcdf.sh "$tstart" "$tend" _GBOFS_stations_hsc.NOW.%H.nc gbofs.nc

It makes you think that this routine could be generalized by specifying the full path with that sort of syntax.

**Appendix B 17 Script Name: GRAPHICS.sh**

Directory Location: /COMF/oqcs/scripts

Technical Contact: Tom Gross Org: NOS/CSDL

Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract: Script to control the creation of all the graphics for a model run. Essential inputs must be defined: $MODELDIR, $MODELWWW, $MODELINFO/stationdata.dat, $MODLEWORK. Copy $MODELINFO/plot_timeseries.ctl, $MODELINFO/plot_field.ctl to $MODELWORK.

The model run NetCDF files must be prebuilt, either directly by the model, or concatenated together using concatnetcdf.sh if necessary. Maybe this function will be put in here.

All control is from the control files which are expected to exist in the MODELDIR/work directory as plot_timeseries.ctl and plot_field.ctl plot_timeseries.ctl is edited with explict directory names to $MODELDIR/work (resolved to non-environment variable format) obs.nc and tide.nc must be in MODELDIR/work along with *.ctl. Output graphics files are put to whereever is specified in the plot control files. Output graphics are copied to the CO-OPS web site mounted to: $MODELWWW.

IDL programs are three stages: plot_timeseries.sh. Sets a couple of IDL parameters and calls IDL; run_plot_timeseries. Defines some internal IDL variables and calls; ngofs_timeseries.pro. The actual lengthy IDL program. These three IDL scripts are all in $OPDSDIR/scripts.

Usage:

Interactively: GRAPHICS.sh "$TIME_NOWCASTSTART" "$TIME_FORECASTEND"

Via cron: Called by MODELCRONRUN.sh

Input Parameters: $1 : "2005 01 25 12 00" Nowcast start time.

$2 : "2005 01 29 12 00" Forcast end time.

Language: Bourne Shell Script

Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| NetCDFgetstation_nwlon_fast.sh | /COMF/oqcs/scripts | Gets data and produces NetCDF file. |
| NetCDFgetstations_astro.sh | /COMF/oqcs/scripts | Makes tides only NetCDF Station file. |
| NetCDFgetstation_currents.sh | /COMF/oqcs/scripts | Gets data and produces NetCDF file. |
| plot_field.sh | /COMF/opds/scripts | IDL script to draw field data. |
| plot_timeseries_cu.sh | /COMF/opds/scripts | IDL script to draw current data. |
| plot_timeseries_wl.sh | /COMF/opds/scripts | IDL script to draw water level data. |

Input Files:

| Name | Directory Location | Description |
|---|---|---|
| stationdata.dat | MODELINFO/ | Used by NetCDFgetstation_nwlon.sh |
| currentsdata.dat | MODELINFO/ | Data file. |
| plot_curr.ctl | MODELINFO/ | If the .ctl file is missing, then graphics |
| plot_timeseries_wl.ctl | MODELINFO/ | program will not be attempted. |
| plot_field.ctl | MODELINFO/ | Control file. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| *.png | User defined | Graphics. |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 18 Control file: plot_field.ctl**

```
% Model Name
MODEL_NAME=Chesapeake Bay Operational Forecast System II
SYS_ACRONYM=CBOFS

% File names
FILE_NOW=/comf/development/COMFgross/ohms/CBOFS/work/fieldsnow.nc
FILE_FORE=/comf/development/COMFgross/ohms/CBOFS/work/fieldsfore.nc
FILE_SHORELINE=/comf/development/COMFgross/ohms/CBOFS/info/shoreline_cbay_medium.
dat

% Output directory name
DIR_OUTPUT=/comf/development/COMFgross/ohms/CBOFS/work/

% List of windows to be plotted. Quoted name string for labeling plots.
% Quoted abbrevation string for created plot file name. Pixel size of Graphic numx numy
% down-left and up-right boundary lat long (decimal degrees)
% Single flag for each variable to be plotted 1(use it) 0(skip it)
% Variable to be plotted: Water Level, Current, Wind, Temperature, Salinity
WN='Chesapeake Bay' 'cb_all' 700 600   36.5 -77.2 39.6 -75.4 5 1 0 1 0 0

% Time window, hindcast duration, forecast duration  hours
TIME_MINMAX=-24  24
% Local Time Zone
TZ=EST5EDT
% Setup data for each plotting variable. Water level
% Auto-Scaling (true) or Fixed Scaling (false) Water Level
WL_AUTOSCALE=True
% Water Level Min Max Range for use with Fixed Scaling (feet)
WL_MINMAX=-2 4
% Units
WL_UNIT=Feet (MLLW)
% Water Level datum adjustment to MLLW: WL_DATUM=MSL-MLLW or MTL-MLLW
% (comment out this part if WL Unit is not MLLW)
WL_DATUM=MTL-MLLW
FILE_DATUM=/comf/development/COMFgross/ohms/CBOFS/info/CBOFS_MLLWdatums.nc

% Currents
% Auto-Scaling (true) or Fixed Scaling (false)
CU_AUTOSCALE=True
% Min Max Range for use with Fixed Scaling
CU_MINMAX=0 3
% Units
CU_UNIT=Knots
CU_LEVEL=0.3,0.6,1.0,1.3,1.6,2.0
```

```
% Wind
% Auto-Scaling (true) or Fixed Scaling (false)
WIND_AUTOSCALE=True
% Min Max Range for use with Fixed Scaling
WIND_MINMAX=0 30
% Units
WIND_UNIT=Knots
WIND_LEVEL=5.,10.,15.,20.,25.,30.

% Temperature
% Auto-Scaling (true) or Fixed Scaling (false)
TEMP_AUTOSCALE=True
% Min Max Range for use with Fixed Scaling
TEMP_MINMAX=0 35
% Units
TEMP_UNIT=Degrees Fahrenheit

% Salinity
% Auto-Scaling (true) or Fixed Scaling (false)
SA_AUTOSCALE=True
% Min Max Range for use with Fixed Scaling
SA_MINMAX=0 45
% Units
SA_UNIT=PSU

% List of city locations for city labels: city name, lat, lon
CITY='Baltimore ' 39.3 -76.69
CITY=' Norfolk' 36.803 -76.22
CITY=' Cambridge' 38.55 -76.05
CITY='Washington ' 38.883 -77.07
```

**Appendix B 19 Control file: plot_timeseries_wl.ctl**

```
% Model Name (use !C to force the displayed line return)
MODEL_NAME=Chesapeake Bay Operational !C!CForecast System
SYS_ACRONYM=CBOFS

% Input File Names
FILE_NOW=/comf/development/COMFgross/ohms/CBOFS/work/stationsnow.nc
FILE_FORE=/comf/development/COMFgross/ohms/CBOFS/work/stationsfore.nc
FILE_OBS=/comf/development/COMFgross/ohms/CBOFS/work/obs.nc
FILE_TIDE=/comf/development/COMFgross/ohms/CBOFS/work/tide.nc

% Output Directory Name
DIR_OUTPUT=/comf/development/COMFgross/ohms/CBOFS/work/

% List of stations to be plotted
% Quoted Name string for labeling plots
% lat long (decimal degrees)
% MLLW Datum Factor to apply to model water levels
%  Old values from the tide tables  CBBT=0.442
% Triplets for each parameter to be plotted  Model, Observation, Tide  1(use it) 0(skip it)
% Different plotting variables specified
% WaterLevelPLot,   Temperature, Salinity
ST='Baltimore Harbor'          'balt' 39.2633 -76.573   0.2499  1 1 1  0 1 0  0 0 0  0
ST='Tochester'                 'tolc' 39.2124 -76.252   0.2591  1 1 1  0 1 0  0 0 0  0
ST='Annapolis Severn River'    'anna' 38.9806 -76.4799  0.2195  1 1 1  0 1 0  0 0 0  0
ST='City of Cambridge, MD.'    'camb' 38.5750 -76.0717  0.3170  1 1 1  0 1 0  0 0 0  0
ST='Solomons Island, MD. '     'solo' 38.3166 -76.4533  0.2347  1 1 1  0 1 0  0 0 0  0
ST='Colonial Beach Pier,MD'    'colo' 38.2516 -76.9600  0.2774  1 1 1  0 1 0  0 0 0  0
ST='Lewisetta, VA        '     'lewi' 37.9967 -76.4633  0.2377  1 1 1  0 1 0  0 0 0  0
ST='Gloucester Point, VA.'     'glou' 37.2467 -76.5000  0.4176  1 1 1  0 1 0  0 1 0  0
ST='Kiptopeke, VA        '     'kipt' 37.1667 -75.9833  0.4450  1 1 1  0 1 0  0 0 0  0
ST='Sewells Point/Hampton Roads'  'hamp' 36.9467 -76.3300  0.4206  1 1 1  0 1 0  0 1 0  0
ST='Chesapeake Bay Bridge Tunnel' 'cbbt' 36.9623 -76.1111  0.4420  1 1 1  0 1 0  0 1 0  1
ST='Thomas Point Light'        'tplm' 38.8983 -76.4366  0.0000  0 0 0  0 0 0  0 0 0  1

% Time window,  hindcast duration , forecast duration  hours
TIME_MINMAX=-24  24
% Local Time Zone
TZ=EST5EDT

% Setup data for each plotting variable
% Water level
% Auto-Scaling (true) or Fixed Scaling (false)  Water Level
WL_AUTOSCALE=True
% Water Level Min Max Range for use with Fixed Scaling (feet)
```

```
WL_MINMAX=-6 6
% Units
WL_UNIT=Feet (MLLW)
% Temperature
% Auto-Scaling (true) or Fixed Scaling (false)

TEMP_AUTOSCALE=True
% Min Max Range for use with Fixed Scaling
TEMP_MINMAX=0 100
% Units
TEMP_UNIT=Fahrenheit Degrees

% Salinity
% Auto-Scaling (true) or Fixed Scaling (false)
SA_AUTOSCALE=True
% Min Max Range for use with Fixed Scaling
SA_MINMAX=0 45
% Units
SA_UNIT=PSU

% Wind
% Auto-Scaling (true) or Fixed Scaling (false)
WIND_AUTOSCALE=True
% Min Max Range for use with Fixed Scaling
WIND_MINMAX=0 40
% Units
WIND_UNIT=Knots
% Wind data plotted frequency, 3 for plotting 1/3 of data points, 5 for 1/5.
WIND_FREQ=1
```

**Appendix B 20 Script Name: hotstart_copy.sh**
Directory Location: /COMF/oqcs/scripts

Technical Contact: Mark Vincent Org: NOS/CSDL
Phone: 301-713-2890 x 151 E-Mail: mark.vincent@noaa.gov
Name: Zack Bronder Org: NOS/CSDL
Phone: 301-713-2890 x 152 E-Mail: zachary.bronder@noaa.gov

Abstract:
This script compares the size (in bytes using du -b) of a hotstart file generated at the end of a model run to the size of a perfect complete hotstart file. If the sizes match, the hotstart file will be copied to the $MODELINIT directory for use in initialzing the next run. This prevents ending up with a corrupted hotstart file during an aborted run or corrupted writing of hotstart. Allowance for copying one additional "optional" file (for example a file with the hotstart times).

Usage: Interactively: NA
Via cron: Called by the MAIN_**OFS.sh script of each model system in
MODULE 5 (RUN MODELS) after the nowcast(s) are run.
If nested models are used it will need to be called twice.
DON'T call after forecast runs since those hotstart files are not used again.

Input Parameters: hotstart_copy.sh testsize hotstart_in hotstart_out optional_in optional_out
testsize = size of a perfect complete hotstart file (in bytes using du -b)
hotstart_in = name of the hotstart file produced by the **OFS in $MODELWORK
hotstart_out = name of the hotstart file (if complete size) copied to $MODELINIT
optional_in = name of an optional file produced by the **OFS in $MODELWORK
optional_out = name of an optional file (i.e. hotstart times) copied to $MODELINIT
$1 size of a complete hotstart file in bytes
$2 hotstart.in file
$3 hotstart.out file
$4 optional *in file (for example a file with the hotstart times)
$5 optional *out file (for example a file with the hotstart times)
Language: Bourne Shell Script
Target Computer: dsofs1.nos-tcn.noaa.gov
Input Files:

| Name | Directory Location | Description |
|---|---|---|
| hotstart_in or equiv. | /COMF/ohms/**ofs/work | the most recent nowcast hotstart file |
| optional_in file | /COMF/ohms/**ofs/work | an optional file with hotstart info. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| hotstart_out or equiv. | $MODELINIT | the hotstart file copied to be used for the next run |
| optional_out file | $MODELINIT | the optional file copied to be used for the next run |

Author Name: Mark Vincent Creation Date: March 17, 2004

**Appendix B 21 Script Name: how_new.sh**

Directory Location: /COMF/oqcs/scripts

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Abstract:

Computes how many hours old the forecast is by comparing
last time in NetCDF file against the $tstart
diffhours = $nctime_end - $tstart
Positive means tstart is before nctime_end
Negative means tstart is after nctime_end
Bad condition would be if tstart is 24 hours after nctime_end = -24
So Corms flags will be r<-16 <y< -8 <g to indicate that forecast too old

Usage: Interactively: how_new.sh tstart file_netcdf
how_new.sh "2003 06 05 12 00" cbbt.nc
Via cron: Called by WINDQCF.sh

Input Parameters: $1: starting time Ex. "2002 01 01 00 00"
$2: the output NetCDF file name Ex. cbbt.nc

Language: Bourne Shell Script

Target Computer: COMF computer. gbofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| datemath | /COMF/oqcs/sorc | Do simple addition, subtraction of dates |
| ncdump | /COMF/oqcs/binlinux | NetCDF dump data to screen tool. |

Input Files:

| Name | Directory Location | Description |
|---|---|---|
| netcdf file | User defined | Output NetCDF file. |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 22 Script Name: how_old.sh**

Directory Location: /COMF/oqcs/scripts

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Abstract:

Computes how many hours old the forecast is by comparing
first time in NetCDF file against the $tstart
diffhours = $nctime_first - $tstart
Positive means tstart is before nctime_first
Negative means tstart is after nctime_first
Bad condition would be if tstart is 24 hours after nctime_first, = -24
So Corms flags will be r<-16 <y< -8 <g to indicate that forecast too old

Usage: Interactively: how_old.sh tstart file_netcdf
how_old.sh "2003 06 05 12 00" cbbt.nc
Via cron: Called by WINDQCF.sh

Input Parameters: Starting time Ex. "2002 01 01 00 00"
The output NetCDF file name Ex. cbbt.nc

Language: Bourne Shell Script

Target Computer: COMF computer. gbofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| datemath | /COMF/oqcs/sorc | Do simple addition, subtraction of dates |
| ncdump | /COMF/oqcs/binlinux | NetCDF dump data to screen tool. |

Input Files:

| Name | Directory Location | Description |
|---|---|---|
| netcdf file | Any | Output NetCDF file. |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 23 Script Name: MAKECORMSFLAGS.sh**

Directory Location: /COMF/oqcs/scripts

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract:

Process the $CORMSLOG file and turn it into the 010 type corms file -- $MODELLOGDIR/cormsflags.010. Then ftp that file to the CORMS Computer along with a zzzz date file to tell it that a new file has arrived and a text file describing the flags Previously this was in each model/scripts directory. It is now oqcs/scripts. It requires only the exported directory names:
$MODELDIR $MODELWORK $MODELWWW and file: $CORMSLOG
Which point to the fixed name file :
$MODELDIR/info/corms_table.txt
Output is to MODELWWW and to archive/CORMSFLAG/%Y%m
Files are copied to:
$MODELDIR/archive/CORMSFLAGS/%Y/%m/%Y%m0%H00cormslog.txt
$MODELWORK/corms_colorflags.txt

Usage: Interactively: MAKECORMSFLAGS.sh "$time_nowcastend"
Via cron: Called by MODELCRONRUN.sh

Input Parameters: $1= "$time_nowcastend"
"2005 01 24 12 00" year month day hour minutes

Language: Bourne Shell Script

Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| CORMSPROCESS.pl | /COMF/oqcs/scripts | CORMS flag processor. |
| Dateformat | /COMF/oqcs/binlinux…sgi | Flexible String builder using dates. |

Output Files:

| Name | Directory Location |
|---|---|
| corms_colorflags.txt | $MODELWWW |
| corms_table.txt | $MODELWWW |
| corms_table.txt | $MODELWWW/zzzz |
| %Y%m0%H00corms_colorflags.txt | $MODELDIR/archive/CORMSFLAGS/%Y%m |
| %Y%m0%H00corms_raw.txt | $MODELDIR/archive/CORMSFLAGS/%Y%m/ |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 24 Script Name: MODELCRONRUN.sh**

```
# ../cbofs2/scripts/CRON_cbofs2.sh
# runs the cbofs model on the gbofs1 machine
#

SETE=/comf/staging/COMF/oqcs/setenvironmentvariables_dsofs1.sh
MODELDIR=/comf/staging/COMF/ohms/CBOFS

# MAIN_INIT_CBOFS2.sh Daily reinitialization: TPLM2, CBBT wind and NWLON water levels
35 13 * * *  source $SETE ; $MODELDIR/scripts/MAIN_INIT_CBOFS2.sh &> \
$MODELDIR/execlog/logcbofsINIT

#
# CBOFSNOW.sh Nowcast and Forecast launches
10 0,6,12,18  * * * source $SETE ; $MODELDIR/scripts/MAIN_CBOFS2.sh &> \
$MODELDIR/execlog/logcbofsMAIN
```

**Appendix B 25 Script Name: NCLwindgetNAMstation.sh**

Directory Location: COMF/oqcs/scripts/
Technical Contact(s): Name: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract:

Reads the NAM full Continental U.S. NetCDF file and interpolates the data to a single lat, lon pair location. This script returns an ASCII file of the TS2 or TS1 type for all the variables in the NetCDF file. It also synthesizes a speed direction file to be compatible with the other wind data sources.

Script for reading NAM NetCDF file. Locate the closest node to lon lat input location

Usage: Interactively: NCLwindgetNAMstation.sh lon lat $STATIONOUTFILEROOT
Notice there are NO quotes around the lon, lat.
NCLwindgetNAMstation.sh -78 36 cbbt
Should create cbbt.nc, cbbtwind.txt, cbbttemp.txt, cbbtpres.txt
Via cron: Called by PRESQCF.sh, WINDQCF.sh.

Input Parameters: lon : longitude
lat : latitude
$STATIONOUTFILEROOT : output file name root.

Language: Bourne Shell Script

Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov.

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| datemath | /COMF/oqcs/sorc | Do simple addition, subtraction of dates. |
| dateformat | /COMF/oqcs/sorc | Flexible String builder using dates. |

Output Files:

| Name | Description |
|---|---|
| $STATIONOUTFILEROOTwind.txt | Ueastward, vnorthward |
| $STATIONOUTFILEROOTspdir.txt | speed, direction from deg North |
| $STATIONOUTFILEROOTtemp.txt | temperature |
| $STATIONOUTFILEROOTpress.txt | pressure |
| $STATIONOUTFILEROOT.nc | A single station NETCDF with time,lat,lon,uwind,vwind,temp,press |

Author Name: Tom Gross Creation Date: Jan, 2005

**Appendix B 26 Script Name: NCLwindgetNAMsub.sh**

Directory Location: COMF/oqcs/scripts/

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Abstract:

Read a subdomain out of the NAM full Continental U.S. NetCDF file. It returns a NetCDF file with time, lat, lon, uwind, vwind, temp, press. The file is much smaller than the original and does not have all the extra meteorological fields that ODAAS provides.
Assumes that the full domain GRIB file has been converted into the local file LocalNAMConus.nc for use by just this sub region extraction.
Example to run from directory work
../scripts/NCLwindgetNAMsub.sh -78 -74 36 40 winds.nc

Usage: Interactively: NCLwindgetNAMsub.sh -78 -74 36 40 winds.nc
Notice there are NO quotes around the lon,lat minimums and maximums.
NCLwindgetNAMsub.sh -76 -75 34 35 "2003 07 08 10 00" junkNAM.nc
Via cron: Called by WINDQCF.sh

Input Parameters: -78 -74 36 40

These longitudes and latitudes will be used to find the upper and lower corners of a square patch in the NAM Lambert projection space. Thus there could be some variation in the other corners from what you might expect, ie points which appear outside your domain or points which should be inside but are excluded. Error on the side of caution and make the min-max's wider than necessary.

Language: Bourne Shell Script

Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| datemath | /COMF/oqcs/sorc | Do simple addition, subtraction of dates. |
| dateformat | /COMF/oqcs/sorc | Flexible String builder using dates. |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 27 Script Name: NetCDFgetstation_currents.sh**

Directory Location: /COMF/oqcs/scripts

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Abstract:

Gets data from the National Ports Database and produces a station NetCDF file. Produces two NetCDF files, _obs and _tide.nc.

Loops through a list of station IDs and names which are contained in the file tationdata.input

stationdata.input has entries like:
8638610 hamp "Hampton Roads Sewells Point"
36 56.8 N 76 19.8 W

Usage: Interactively: NetCDFgetstation_currents.sh currents.dat "2000 12 31 00 00" \
"2000 12 31 00 59" 0.1 currentsobs.nc currentspred.nc
Via cron: Called by GRAPHICS.sh

Input Parameters:
$1: current data file name
$2: starting time
$3: ending time
$4: time interval
$5: currents observation data NetCDF file name
$6: currents prediction data NetCDF file name

Language: Bourne Shell Script

Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| get_data_npdb_currents.sh | /COMF/oqcs/scripts | Accesses PORTS Database to get currents |
| pred_ngofs.x | /COMF/oqcs/binlinux | Makes multiple years prediction |
| catstationcurrnetcdf.x | /COMF/oqcs/binlinux | Adjusts several obs data onto a single time line |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 28 Script Name: NetCDFgetstation_nwlon_fast.sh**

Directory Location: /COMF/oqcs/scripts
Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Abstract:

Gets data from NWLON and produces a station NetCDF file. Produce two NetCDF files, rootfilename_obs and rootfilename_tide.nc. Use WLQCF.sh to grab tides. Use get_data_nwlon_db.sh to get water levels and so on.

Usage: Interactively: give the starting and ending times, the stationdata.input file name and the output NetCDF file name:

```
NetCDFgetstation_nwlon_fast.sh  stationdata.input "2002 01 01 00 00" \
  "2002 01 12 12 00" obscbbay.nc
```

Via cron: Called by Graphics.sh.

Input Parameters: $1: station data input data file name
$2: Starting time
$3: Ending time
$4: Output NetCDF file name
Language: Bourne Shell Script
Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| WLQCF.sh | /COMF/oqcs/scripts | Grabs Water Level Datal. |
| get_data_nwlon_db.sh | /COMF/oqcs/scripts | Reads NWLON data using SYBASE interface. |
| SALINITY.pl | /COMF/oqcs/scripts | Get conductivity and temperature. |
| datemath | /COMF/oqcs/binlinux | Do simple addition, subtraction of dates. |
| catstationobsnetcdf.x | /COMF/oqcs/binlinux | Reads in several files of observation data. |
| fillnan.x | /COMF/oqcs/binlinux | Fill nan to the non-exsits date. |

Author Name: Tom Gross Creation Date: 2003
Remarks:

The flags are not tested on the NWLON water levels returned by get_data_nwlon_db.sh. So slightly bad data is not left out and IS plotted. If the flags were tested more data would be left out, but maybe too much.

```
get_data_nwlon_db.sh $stnid "$begindate" "$enddate" WL DAT
feb 3, 2004  Turn off the flag tests
#awk ' $7 == "0" && $8 == "0" && $9 == "0" && $10 == "0" && $11 == "0" {print }'\
DAT |  cat DAT | fillnan.x "$begindate" "$enddate" $DT " -99999.000 0 0 0 0 0" | \
awk '{printf("0 0 0 0 0 0 0.000000\n", $1, $2, $3, $4, $5, $6)}' > \
SCRATCHDIR"/"$stationname"_obs"
```

**Appendix B 29 Script Name: NetCDFgetstations_astro.sh**

Directory Location: /COMF/oqcs/scripts

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Abstract:

NetCDFgetstations_astro.sh stationfile gets data from NWLON and produces a station NetCDF file. Produces two NetCDF files, _obs and _tide.nc

Use WLQCF.sh to grab obs and tides. Specify similar parameters to WLQCF.sh. Except the stationid is a file of IDs, names, lat, lon. Loop through a list of station IDs and names which are contained in the file stationdata.input. stationdata.input has entries like:

8638610 hamp "Hampton Roads Sewells Point"
36 56.8 N 76 19.8 W

Usage: Interactively: NetCDFgetstations_astro.sh stationfile tstart tend DT ncfilename
NetCDFgetstations_astro.sh stationdata.input "2002 01 01 00 00"\
"2002 01 12 12 00" cbbaytide.nc
Via cron: Called by GRAPHICS.sh

Input Parameters: stationfile : station input data file name.
tstart : starting time.
tend : ending time.
DT : time interval.
ncfilename : output NetCDF file name.

Language: Bourne Shell Script

Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| WLQCF.sh | /COMF/oqcs/scripts | Grab Water Level Data. |
| mktemp | /COMF/oqcs/binlinux..sgi | Makes a temporary unique filename. |
| dateformat | /COMF/oqcs/binlinux..sgi | Flexible String builder using dates. |
| obstidenetcdf.x | /COMF/oqcs/binlinux | Construct tides.nc, obs.nc station files. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| _obs.nc | User defined | Observation NetCDF file. |
| _tide.nc | User defined | Tidal NetCDF file. |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 30 Script Name: notbracket.pl**

Directory Location: /COMF/oqcs/scripts

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Abstract:

Perl script to take all "non" html lines out of a file.
Obviously this can use some work.
But it works for the few applications we have.
Print the lines without the < html > tags

Usage:

Interactively: cat $file | notbracket.pl > $outfile
perl $OQCSDIR/scripts/notbracket.pl $WGETOUT |tr "/:" " " > $PERLED

Via cron: Script is called by tide_read_nwlon.sh

Input Parameters: A file.

Language: Perl Script

Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Input Files:

| Name | Directory Location | Description |
|---|---|---|
| $File | User given | Usually an html file. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| $outputfile | User given | Get rid of html special character. |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 31 Program Name: OFS_CONTROL.sh**

Location: COMF/oqcs/scripts/

Technical Contact: Zack Bronder — Org: NOS/CSDL
Phone: 301-713-2890 x152 — E-Mail: Zachary.Bronder@noaa.gov
Mark Vincent — Org: NOS/CO-OPS
Phone: 301-713-2890 x150 — E-Mail: Mark.Vincent@noaa.gov

Abstract: OFS_CONTROL.sh is a standard part of COMF (Coastal Ocean Modeling Framework). This script is used to determine whether or not a COMF coastal/estuarine model should be launched. If appropriate, it will run the model.

Usage: Interactively: OFS_CONTROL.sh
Automatically: OFS_CONTROL.sh will be called by COMF model scripts with the naming convention MAIN_??OFS.sh, where ?? stands for a two or three characters model abbreviation such as CB (Chesapeake Bay), NY, GB, TB, SJR, and so on.

Input Parameters: None
Language: Bourne Shell Script
Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Input Files:

| Name | Location | Description |
|---|---|---|
| ofs_control_prevented | $MODELINIT | Exists if parent process is prevented. |

Author Name: Zack Bronder — Creation Date: October 1, 2003

Remarks:

OFS_CONTROL.sh was developed on gbofs1.nos.noaa.gov. It was designed to be used by NOS estuarine/coastal hydrodynamic models within COMF. It can actually be called by any program to prevent it from running if other processes with the same name are running, provided that $MODELDIR variable is set to where the prevented status file would be located and is exported to this script.

Assign variables.
Get the name of this script's parent process.
Get the number of processes on the system with the same name as $parent_name.
Print variable values.
Check if parent process should continue to run.
There can be only one.
Check if prevented_file exists.

**Appendix B 32 Script Name: pres_read_nwlonweb.sh**

Directory Location: /COMF/oqcs/scripts

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Aijun Zhang Org: NOS/CSDL
Phone: 301-713-2809x113 E-Mail: aijun.zhang@noaa.gov

Abstract:

Grabs the air temperature data off the NWLON web site.
It depends upon this line:

```
echo "http://www.co-ops.nos.noaa.gov/cgi-bin/co-ops_qry_direct.cgi?\
stn=$stnid&dcp=1&ssid=F1&pc=W1&datum=NULL&unit=0&bdate=$bdate\
&edate=$edate&date=3&shift=0&level=1&form=0&host=&addr=10.60.5.243\
&data_type=pan&format=View+Data" > $REQUESTGET
```

As with all screen scrapers if the CO-OPS changes this reference,
then this program will crash.
Output file has date, forecasthour, waterlevel
(nwlon, pressure forecasthour ==0 )

```
y   m d h m   fh   pres
2005 02 15 00 00 00   1017.5
2005 02 15 00 06 00   1017.4
```

Usage: Interactively: pres_read_nwlonweb.sh stationid startdate enddate outputfilename
Via cron: Called by PRESQCF.sh.

Input Parameters:
station id Ex. 8638610
start date Ex. "2002 12 10 00 00"
end date Ex. "2002 12 12 12 00"
output file name Ex. pres8638610.txt

Language: Bourne Shell Script
Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| dateformat | /COMF/oqcs/bin...sorc | Flexible String builder using dates. |
| mktemp | /COMF/oqcs/bin../sorc | Makes a temporary unique filename. |
| wget | /COMF/oqcs/binsgi | Web grabber. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| $4 | Depend on requests. | y m d h m fh pres |

Author Name: Aijun Zhang Creation Date: Jan. 15, 2005

**Appendix B 33 Script Name: PRESQCF.sh**

Directory Location: /COMF/oqcs/scripts
Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Abstract: Pressure read, QC. Format script.
Purpose: Return a QC'd, gap filled time series of PRESSURE data. Produces $QCEDfile from tstart to tend at DT(seconds) spacing.
y m d h min fh pressure
2002 12 29 12 30 0 20.5678
Returns surface air pressure, mbars.
Standard T1 file:
y m d h min fh pressure(mbar)
2002 12 29 12 30 0 1000.2

Usage: Interactively:
PRESQCF.sh 8863863 NWLON "2003 03 22 00 00" "2003 03 27 12 00" 0.10 cbbtpress.out
PRESQCF.sh stationid database tstart tend DT QCEDfile
Via cron: Called by MODELCRONRUN.sh

Database: NWLON where available. Nowhere near all NWLON stations have pressure.
NAMSTATION T1 file for a lat,long location from the NAM forecast
To get a NetCDF field of pressure use WINDQCF.sh

Input Parameters:
stationid : station ID or field range(8863863, or "35.5 -78.3")
database : database name(NWLON, NAMSTATION)
tstart : starting time ("2005 03 22 00 00")
tend : ending time ("2005 03 28 00 00")
DT : time interval, in hours 0.10 = 6min
QCEDfile : output file name (ASCII file)

Language: Bourne Shell Script
Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov.

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| get_data_nwlon_db.sh | /COMF/oqcs/scripts | Reads NWLON data using SYBASE interface. |
| cormspercent.sh | /COMF/oqcs/scripts | Calculates percentage for CORMS flags. |
| dateformat | /COMF/oqcs/sorc | Flexible String builder using dates. |
| gapfill.f | /COMF/oqcs/sorc | Gap fills with linear ramps. |
| mktemp.c | /COMF/oqcs/sorc | Makes a temporary unique filename. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| QCEDfile | User defined | y m d h min fh pressure(mbar) |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 34 Program Name: PURGE.sh**

Technical Contact: Zack Bronder Org: NOS/CO-OPS
Phone: 301-713-2890 x152 E-Mail: Zachary.Bronder@noaa.gov
Mark Vincent Org: NOS/CO-OPS
Phone: 301-713-2890 X150 E-Mail: Mark.Vincent@noaa.gov

Location: $OQCSDIR/scripts/

Abstract:

A standard part of COMF, PURGE.sh is used to delete archived files from NOS operational coastal forecast systems in order to conserve disk space. It is a counterpart to ARCHIVE.sh and ARCHIVE_GRAPHICS.sh, in that it removes files that these scripts have archived.

Usage: Interactively: PURGE.sh
Automatically: PURGE.sh will be called by MAIN_??OFS.sh and run as a cron job.

Input Parameters: PURGE.sh uses a control file, $MODELINFO/PURGE.ctl, to direct the purging. This control file has entries with three fields per line: directory string, file string, and days string. The directory string corresponds to the subdirectory below $ARCHIVEDIR, which is named for a type of archived file (NetCDF, graphics, etc). The file string corresponds to the name(s) of the file(s) that will be purged. The file string may contain wildcards. The days string refers to how many days old the files to be purged are. This is used to construct the names of files to be purged. The file names include a date string at the beginning, followed by an underscore, then model name, underscore, file type, and possibly an extension at the end of the name.

Language: Bourne Shell Script

Programs Called:

| Name | Location | Description |
|---|---|---|
| dateformat | /COMF/oqcs/binlinux/dateformat | A C program that formats date strings. |

Input Files:

| Name | Description |
|---|---|
| $MODELINFO/PURGE.ctl | Controls which files PURGE.sh will remove. |
| $ARCHIVEDIR/$subdirectory/$rmfiles | Files that will be removed. |

Libraries Used: None

Author: Zack Bronder Date: September 14, 2004

Remarks: This script needs the following to be defined:
$ARCHIVEDIR, $MODELINFO

**Appendix B 35 Script Name: read_ndbc_archive.sh**

Directory Location: /COMF/oqcs/scripts

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Abstract: Gets data from COMF/oqcs/archive/ndbc, and produces an ASCII file.
The archives keep hourly data of most variables.
Available : WIND WD WSPD GST WVHT DPD.
APD MWD BAR ATMP WTMP DEWP VIS TIDE
(WIND will give back the composite of WSPD and WD) like wind_read_ndbc.sh
WIND Output file has date, forecasthour, windspeed, winddir
(ndbc, nwlon, tide forecasthour ==0 )
y m d h m fh speed dir
2002 12 30 12 00 0 .5678 270
or
Output file has date, forecasthour, variable
(ndbc, nwlon, tide forecasthour ==0 )
y m d h m fh GST
2002 12 30 12 00 0 4.7

Usage: Interactively: read_ndbc_archive.sh stationid datavariable startdate enddate outputfilename
Via cron: Called by WLQCF.sh

Input Parameters:
station ID Ex TPLM2
data variable Ex GST
starting time Ex "2003 01 09 00 00"
ending time Ex "2003 01 12 12 00"
outputfilename Ex GSTTPLM2.tx

Language: Bourne Shell Script

Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| dateformat | /COMF/oqcs/sorc | Flexible String builder using dates. |
| datemath | /COMF/oqcs/sorc | Do simple addition, subtraction of dates. |
| wget | /COMF/oqcs/binlinux | Web Grabber. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| $5 | user defined | Text data file. |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 36 Script Name: READUSGS.pl**

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov
Directory Location: /COMF/oqcs/scripts

Abstract: This Perl script parses the USGS web page(grabbed using river_read_usgs.sh) to isolate and return one or more of these data variables:
TEMP 00010 - TEMPERATURE, WATER (DEG. C)
COND 00095 - SPECIFIC CONDUCTANCE (MICROSIEMENS/CM AT 25 DEG. C)
DISCHARGE 00060 - DISCHARGE, CUBIC m PER SECOND
GAGE 00065 - GAGE HEIGHT, m
Returns -9999.00000 if the station doesn't have the data type.
Converstion from feet to MKS is done here.

Usage:
Interactively: READUSGS.pl $INPUTFILENAME "variable list" $OUTPUTFILENAME
Via cron: The outputfile is simple ASCII file (with only as many v2,v3,v4 as requested):
y m d h m 00 v1 v2 v3 v4

Input Parameters: The variable list should be something like:
"DISCHARGE"
"TEMP"
"TEMP COND"
"TEMP DISCHARGE GAGE COND"
"DISCHARGE GAGE TEMP COND"
Language: Bourne Shell Script
Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Iutput Files:

| Name | Directory Location | Description |
|---|---|---|
| $INPUTFILENAME | user defined | data file. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| $OUTPUTFILENAME | user defined | simple ascii<br>with as many v2,v3,v4 as requested<br>y m d h m 00 v1 v2 v3 v4 |

Author Name: Tom Gross Creation Date: 2003

Remarks:
This is a real tour de force in hash table redirection sub-scripting. Study the line:
printf(OUTFILE " 0.000000",$VAL[$POS{$VARDD{$VARIABLELIST[$i]}}]
*$MKS{$VARIABLELIST[$i]});

**Appendix B 37 Script Name: river_read_usgs.sh**

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Directory Location: /COMF/oqcs/scripts

Abstract: Gets data from the USGS Real-time River flow web page:
http://waterdata.usgs.gov/md/nwis/uv?01578310
Requests tab separated data and you will see the source file.
There is no choice about times on this web page, so this only
gives you the last SEVEN days of data.
The script decodes these files to grab the different data
types which might be available. Not all stations have the
same data (or in the same order.) Possible choices are:
TEMP TEMPERATURE, WATER (DEG. C)
COND SPECIFIC CONDUCTANCE (MICROSIEMENS/CM AT 25 DEG. C)
DISCHARGE DISCHARGE, CUBIC FEET PER SECOND
GAGE GAGE HEIGHT, FEET
The requested page is sent to READUSGS.pl to parse out the data type requested.
Produces an ASCII file, capable of returning any data variables from any river station.
Returns an ASCII file like:
2003 04 17 00 00 0 6.390000 63700.000000
2003 04 17 00 30 0 6.380000 63600.000000

Usage: Interactively: river_read_usgs.sh stationid listvar startdate enddate outputfilename
Via cron: Called by RIVERQCF.sh, SALTQCF.sh, TEMPQCF.sh, WLQCF.sh

Input Parameters: stationid number Ex.01570500
list of variables Ex. "GAGE DISCHARGE"
starting time Ex "2003 04 01 00 00"
ending time Ex "2003 04 26 12 00"
output file name Ex riv.dat

Language: Bourne Shell Script

Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| READUSGS.pl | /COMF/oqcs/scripts | Parses out the data from USGS river web page. |
| mktemp.c | /COMF/oqcs/sorc | Makes a temporary unique filename. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| riv.dat | User given | 2003 04 17 00 00 0 6.390 63700.00 |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 38 Script Name: river_read_usgsarchive.sh**

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Directory Location: /COMF/oqcs/scripts

Abstract: Gets data from the USGS Archive web site
Daily data only: http://waterdata.usgs.gov/md/nwis/uv?01578310
Request tab separated data and you will see the source file. There is no choice about times on this web page, so this only gives you the last SEVEN days of data.
The script decodes these files to grab the different data types which might be available. Not all stations have the same data (or in the same order.) Possible choices are:
TEMP TEMPERATURE, WATER (DEG. C)
COND SPECIFIC CONDUCTANCE (MICROSIEMENS/CM AT 25 DEG. C)
DISCHARGE DISCHARGE, CUBIC FEET PER SECOND
GAGE GAGE HEIGHT, FEET
The requested page is sent to READUSGS.pl to parse out the data type requested.
Produces an ASCII file like:
2001 04 01 00 00 111000.0
2001 04 02 00 00 107000.0
Daily mean streamflow value, in cubic-meter per-second

Usage: Interactively: river_read_usgsarchive.sh stationid listvar startdate enddate outputfilename
Via cron: Called by RIVERQCF.sh.

Input Parameters:
stationid number Ex.01570500
list of variables Ex. "GAGE DISCHARGE"
starting time Ex "2003 04 01 00 00"
ending time Ex "2003 04 26 12 00"
output file name Ex riv.dat

Language: Bourne Shell Script

Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| dateformat | /COMF/oqcs/bin...sorc | Flexible String builder using dates. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| text file | As input given | 2001 04 01 00 00 111000.0 |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 39 Script Name: river_read_usgsmysql.sh**

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Directory Location: /COMF/oqcs/scripts

Abstract: Gets data from the USGS and produces an ASCII file.
Capable of returning any data variables from any river station.
Returns an ASCII file like:
2003 04 17 00 00 0 6.390000 63700.000000
2003 04 17 00 30 0 6.380000 63600.000000
Variables:
TEMP TEMPERATURE, WATER (DEG. C)
COND SPECIFIC CONDUCTANCE (MICROSIEMENS/CM AT 25 DEG. C)
DISCHARGE DISCHARGE, CUBIC FEET PER SECOND
GAGE GAGE HEIGHT, FEET
mysql varient only does DISCHARGE, but it requires DISCHARGE be present

Usage: Interactively: river_read_usgsmysql.sh stationid listvar startdate enddate outputfilename
Via cron: Called by RIVERQCF.sh

Input Parameters: stationid number Ex. 01570500
list of variables Ex. "GAGE DISCHARGE"
starting time Ex "2003 04 01 00 00
ending time Ex "2003 04 26 12 00"
output file name Ex. riv.da

Language: Bourne Shell Script
Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov.

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| dateformat | /COMF/oqcs/sorc | Flexible String builder using dates. |
| mktemp.c | /COMF/oqcs/sorc | Makes a temporary unique filename. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| outputfilename | User defined | 2003 04 17 00 00 0 6.390000 63700.000000 |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 40 Script Name: RIVERQCF.sh**

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Directory Location: /COMF/oqcs/scripts/

Abstract: Returns river discharge in volume per time, i.e., m^3/sec.
Standard TS1 file:
y m d h min fh discharge(m3/sec)
2002 12 29 12 30 0 200.2
RIVER READ, QC, Format script
Purpose:
Return a QC'd, gap filled time series of RIVER DATA
Produces $QCEDfile from tstart to tend at DT(seconds) spacing
y m d h min fh discharge(m3/sec)
2002 12 29 12 30 0 200.2
Reads the observation for a station from
the selected database. Then calls gapfill.f to do some
editing and intelligent filling

Usage: Interactively: RIVERQCF.sh stationid database tstart tend DT QCEDfile
Via cron: Called by MODELCRONRUN.sh

Input Parameters:
stationid : station ID (8863863)
database : database name(NWLON, NDBC)
tstart : starting time ("2005 03 22 00 00")
tend : ending time ("2005 03 28 00 00")
DT : time interval, in hours 0.10 = 6min
QCEDfile : output file name (ASCII file)

Language: Bourne Shell Script

Target Computer: COMF computers, such as dsofs1.nos-tcn.noaa.gov.

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| rive_read_usgs.sh | /COMF/oqcs/scripts | Screen Scrapper for USGS river discharge. |
| cormspercent.sh | /COMF/oqcs/scripts | Calculates percentage for CORMS flags. |
| datemath | /COMF/oqcs/sorc | Do simple addition, subtraction of dates. |
| dateformat | /COMF/oqcs/sorc | Flexible String builder using dates. |
| wlgapfill.f | /COMF/oqcs/sorc | Water Level edit and gap filler. |
| mktemp.c | /COMF/oqcs/sorc | Makes a temporary unique filename. |
| rive_read_usgsmysql.sh | /COMF/oqcs/scripts | Gets data from the USGS, produces a ascii file. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| QCEDfile | User defined | y m d h min fh discharge(m3/sec) |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 41 Script Name: setenvironmentvariables.sh**

```
# Make a copy of setenvironmentvariables.sh to setenvironmentvariables_dsofs1_username.sh
# Edit the COMFDIR and MODELDIR variables.
# Every thing else should be specific to the dsofs1 machine.
# Do not cvs add your individual setenvironmentvariables_dsofs1_username.sh
# An example bash script setting the directory names as environment variables for use throughout
# the modeling system.
# Use these environment variables whereever there is a question as to where something resides in
# the system. If these are set correctly then the system becomes wonderfully relocatable by only
# changing this file.

COMFDIR=/comf/staging/COMF
MODELDIR=$COMFDIR/ohms/CBOFS
ODAASDIR=/comf/odaas

# alter for sgi or linux
OQCSBIN=$COMFDIR/oqcs/binlinux
NCARG_ROOT=$COMFDIR/oqctools/ncarglinux

#  lf95 libraries needed for executables
#  bassmmap default is to have this in .login, which is not called by crontab
# source /usr/local/lf9562/bash_setup
LD_LIBRARY_PATH=/usr/local/lf9562/lib:$LD_LIBRARY_PATH
PFDIR=/usr/local/lf9562/bin
WISK=/usr/local/lf9562
export LD_LIBRARY_PATH PFDIR WISK

# Force a simple PATH
PATH=$NCARG_ROOT/bin:$PFDIR:$WISK:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:.

# SYBASE for use of get_data_nwlon_db.sh
# It be installed by just doing a directory copy.
#  Check out $SYBASE/interfaces
export SYBASE=/opt/sybase-12.5
PATH=$SYBASE/OCS/bin:$PATH
export LANG=C
export NETCDF_ROOT=$COMFDIR/oqctools/netcdflinux

OQCSDIR=$COMFDIR/oqcs
OPDSDIR=$COMFDIR/opds
CORMSLOG=/dev/null
PATH=$PATH:$OQCSBIN:$OQCSDIR/scripts
export PATH COMFDIR OQCSDIR OQCSBIN ODAASDIR OPDSDIR CORMSLOG
export OPDSMATLAB MODELDIR
export NCARG_ROOT
```

**Appendix B 42 Script Name: SALINITY.pl**

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Directory Location: /COMF/oqcs/scripts

Abstract: A perl script used to convert a file containing water conductivity and temperature into just Salinity.
Assumes the Pressure = 0.00.
Uses snippets of code from the matlab toolbox "seawater"
http://www.marine.csiro.au/~morgan/seawater/
It is based on the Unesco equations:
http://www.ices.dk/ocean/procedures/standard_seawater.htm
Check against this handy web based salinity calculator
http://ioc.unesco.org/oceanteacher/resourcekit/M3/Converters/SeaWaterEquationOfState/
Sea%20Water 0.000000E+00quation 0f%20State%20Calculator.htm
It expects the conductivity to be in milli Siemens/cm. Those are the units which are about 1/10 the ppt, i.e., the conductivity at 35ppt, 15C = 4.2914 milli Siemens/cm. Pressure is assumed to be 0, Sea Surface data only. However the pressure correction is slight until you get past 100m or more. The conductivity is temperature corrected by this routine. So it requires the water temperature. This is incompatible with the USGS conductivities which are specific, meaning that they are converted to conductivity at 15C.

Usage: Interactively: SALINITY.pl "$CDAT" "$CDAT2"
Via cron: Called by NetCDFgetstation_nwlon_fast.sh
Called by SALTQCF.sh

Input Parameters:
"$CDAT": Input file, has y m d h m fh conductivity temperature
"$CDAT2": Output file, has y m d h m fh salinity

Language: Bourne Shell Script
Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov.

Input Files:

| Name | Directory Location | Description |
|---|---|---|
| $CDAT | User defined | y m d h m fh conductivity temperature |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| "$CDAT2" | User defined | y m d h m fh salinity |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 43 Script Name: SALTQCF.sh**

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Directory Location: /COMF/oqcs/scripts

Abstract: Returns salinity in TS1 format.
This reads both the water temperature and conductivity.
They are converted to salinity using SALINITY.pl
However NWLON and USGS databases use specific and non-specific conductivity.
So this program is not yet complete for USGS.
Standard TS1 file:
y m d h min fh Salinity (ppt)
2002 12 29 12 30 0 35.3
Reads the observation for a station from the selected database. Then calls gapfill.f to do some editing and intelligent filling.

Usage: Interactively: SALTQCF.sh stationid database tstart tend DT QCEDfile
Via cron: Called by MODELCRONRUN.sh

Input Parameters:
$1 : station id 8863863
$2 : database name NWLON or USGS
$3 : starting time 2005 03 27 12 00
$4 : ending time 2005 03 27 12 00
$5 : time interval. 0.1(hour = 6 minutes)
$6 : output QCF file name. cbbtsalt.dat

Language: Bourne Shell Script
Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov
Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| get_data_nwlon_db.sh | /COMF/oqcs/scripts | Reads data using the SYBASE interface. |
| river_read_usgs.sh | /COMF/oqcs/scripts | Screen Scrapper for USGS river discharge. |
| cormspercent.sh | /COMF/oqcs/scripts | Calculates percentage for CORMS flags. |
| SALINITY.pl | /COMF/oqcs/scripts | Gets Salinity by conductivity, temperature. |
| datemath | /COMF/oqcs/sorc | Do simple addition, subtraction of dates. |
| gapfill.f | /COMF/oqcs/sorc | Gap fills with linear ramps. |
| mktemp.c | /COMF/oqcs/sorc | Makes a temporary unique filename. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| QCF | Given by user | Output TS1 file. |

Author Name: Tom Gross Creation Date: 2003

Remarks: Database:
NWLON where available.
USGS but formulas for conversion might still be broken for nearly fresh water.

**Appendix B 44 Script Name: temp_read_ndbc.sh**

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Directory Location: /COMF/oqcs/scripts

Abstract: temp_read_ndbc.sh gets data from NDBC and produces an ASCII file.
Although this does sort for the date range, it does not go back in time more than one month, i.e., it reads the realtime web site.
Gets data from the NDBC web site.
This actually gets any of the variables:
AT Air Temperature Centigrade
WT Water Temperature Centegrade
AP Air Pressure mbars 1022.6
Output file has date, forecasthour, temperature
(ndbc, nwlon, tide forecasthour ==0 )
y m d h m fh temp
2002 12 30 12 30 0 25.5

Usage: Interactively: temp_read_ndbc.sh stationid startdate enddate sensor outputfilename
Via cron: Called by TEMPQCF.sh

Input Parameters: Station name Ex. TPLM2
starting time Ex. "2003 03 09 00 00"
ending time Ex. "2003 03 12 12 00"
output file name Ex. tempTPLM2.txt
sensor Ex. AT, air temperature

Language: Bourne Shell Script

Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov.

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| dateformat | /COMF/oqcs/sorc | Flexible String builder using dates. |
| mktemp.c | /COMF/oqcs/sorc | Makes a temporary unique filename. |
| wget | /COMF/oqcs/binlinux | Request data from WWW web. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| outputfilename | Given by user | year m d h m fh temp |

Author Name: Tom Gross Creation Date: 2003

Remarks:

Gets data from the NDBC web site. Really only tested on the CMAN stations, but ought to work for the buoys also. Their web page has fixed file names for the data files, so this works nicely:
wget http://www.ndbc.noaa.gov/data/realtime/$stnid.txt -O $WGETOUT -o $WGETLOG
But a limited amount of data is available. Only the most recent 45 days
As a screen scraper this is susceptible to changes. The format of the downloaded files has changed in the past.

**Appendix B 45 Script Name: TEMPQCF.sh**

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Directory Location: /COMF/oqcs/scripts

Abstract: Return a QC'd, gap filled time series of air or water temperature data.
Produces $QCEDfile from tstart to tend at DT (seconds) spacing. Standard TS1 file:
y m d h min fh temp
2002 12 29 12 30 0 15.2 (Celsius)
Reads the observation for a station from the selected database. Then calls qcfill.f to do some editing and intelligent filling

Usage: Interactively: TEMPQCF.sh stationid database SENSOR tstart tend DT QCEDfile
Via cron: Called by MODELCRONRUN.sh
Different from others:
There are a number of Different Temperatures available.
Specify which one using the SENSOR variable:
Also requires code for
AT Air Temperature
WT Water Temperature
Later we need to do:
WTS Surface Water Temperature
WTB Bottom Water Temperature

Input Parameters:
stationid : station ID (8863863)
database : database name(NWLON, NDBC)
SENSOR : temperature variables name(air temperature, AT, ect.)
tstart : starting time ("2005 03 22 00 00")
tend : ending time ("2005 03 28 00 00")
DT : time interval, in hours 0.10 = 6min
QCEDfile : output file name (ASCII file)

Language: Bourne Shell Script

Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov.

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| get_data_nwlon_db.sh | /COMF/oqcs/scripts | Reads data using the SYBASE interface. |
| river_read_usgs.sh | /COMF/oqcs/scripts | Screen Scrapper for USGS river discharge. |
| cormspercent.sh | /COMF/oqcs/scripts | Calculates percentage of data for CORMS flags. |
| temp_read_ndbc.sh | /COMF/oqcs/scripts | Screen Scrapper for NDBC water temperature. |
| datemath | /COMF/oqcs/sorc | Do simple addition, subtraction of dates. |
| gapfill.f | /COMF/oqcs/sorc | Gap fills with linear ramps. |
| mktemp.c | /COMF/oqcs/sorc | Makes a temporary unique filename. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| QCEDfile | Given by user | y m d h min fh temp |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 46 Script Name: tide_read_nwlonweb.sh**

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Directory Location: /COMF/oqcs/scripts

Abstract: Gets data from NWLON predicted tide data and produces an ASCII file
Uses a wget to go through the NWLON predictions page CGI
Use a loop to get more than 30 days data. Web site only offers 30 days data access.
Output file has date, forecasthour, water level
(nwlon, tide forecasthour ==0 )
y m d h m fh tide
2002 12 30 12 30 0 .5678
2005 01 31 00 00 00 0.0080
2005 01 31 00 06 00 0.0140

Usage: Interactively: tide_read_nwlonweb.sh stationid startdate enddate outputfilename
Via cron: Called by WLQCF.sh

Input Parameters: station id Ex. 8638610
start date Ex. "2005 01 01 00 00"
end date Ex. "2005 01 12 12 00"
output file name Ex. tide8638610.txt

Language: Bourne Shell Script
Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| dateformat | /COMF/oqcs/binlinux | Flexible String builder using dates. |
| mktemp | /COMF/oqcs/binlinux | Makes a temporary unique filename. |
| wget | /COMF/oqcs/binsgi | Web Grabber. |
| notbracket.pl | /COMF/oqcs/scripts | Perl script to help screen scrapping. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| tide8638610.txt | Given by user | y m d h m fh tide |

Author Name: Hong Lin Creation Date: 02-04-2005

**Appendix B 47 Script Name: wind_read_ndbc.sh**

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Directory Location: /COMF/oqcs/scripts

Abstract:

Gets data from the NDBC web site and produces an ASCII file. Really only tested on the CMAN stations, but ought to work for the buoys also.
Their web page has fixed file names for the data files so this works nicely:
wget http://www.ndbc.noaa.gov/data/realtime/$stnid.cwind -O $WGETOUT -o $WGETLOG
But a limited amount of data is available, only the most recent 45 days.
As a screen scraper this is susceptible to changes. The format of the downloaded files has changed in the past.
These files have
YYYY MM DD hh mm DIR SPD GDR GSP GMN
and are listed in upside down order. They are converted to
y m d h m 0 speed dir
Called from WINDQCF.sh which converts speed direction to Ueast, Vnorth, inside the Fortran program wind_QC_station_gapfill.f
Although this does sort for the date range, it does not go back in time more than one month. ie it reads the realtime web site.
Output file has date, forecasthour, windspeed, winddir
(ndbc, nwlon, tide forecasthour ==0 )
y m d h m fh wl
2002 12 30 12 30 0 .5678

Usage: Interactively: wind_read_ndbc.sh stationid startdate enddate outputfilename
Via cron: Called by WINDQCF.sh

Input Parameters: station id Ex. TPLM2
start date Ex. "2003 01 09 00 00"
end date Ex. "2003 01 12 12 00"
output file name Ex. windTPLM2.txt)

Language: Bourne Shell Script

Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| dateformat | /COMF/oqcs/sorc | Flexible String builder using dates |
| mktemp.c | /COMF/oqcs/sorc | Makes a temporary unique filename. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| $5 | user defined | ASCII data file. |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 48 Script Name: wind_read_nwlonweb.sh**

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Aijun Zhang Org: NOS/CSDL
Phone: 301-713-2809x113 E-Mail: hong.lin@noaa.gov

Directory Location: /COMF/oqcs/scripts

Abstract:

Grabs the wind speed data off the NWLON web site. Retrieve eteorological Oceanographic Data web page. This uses a screen scraper which directly calls the CGI used to fill in the data from
http://co-ops.nos.noaa.gov/data_retrieve.shtml?input_code=101000111pan
This is an web version and backup to get_data_nwlon_db.sh. It depends upon this line: echo "http://www.co-ops.nos.noaa.gov/cgi-bin/co-ops_qry_direct.cgi?\
stn=$stnid&dcp=1&ssid=C1&pc=W1&datum=NULL&unit=0&bdate=$bdate&edat
e=$edate\&date=3&shift=0&level=1&form=0&host=&addr=10.60.5.243&data_typ
e=pan\&format=View+Data" > $REQUESTGET
Now have a file with stuff like:
Station D SE Date Time WS WD WG X R
8638863 1 C1 2005/02/15 00:00 8.4 155.0 9.2 0 0
Output look like
year mon dd hh mi fh ws wd
2005 02 15 00 00 00 8.4000 155.0000
2005 02 15 00 06 00 8.0000 160.0000
As with all screen scrapers if the CO-OPS changes this reference, then this program will crash.

Usage: Interactively: wind_read_nwlonweb.sh stationid startdate enddate outputfilename
Via cron: Called by WINDQCF.sh

Input Parameters: station id Ex. 8638610
start date Ex. "2002 12 10 00 00"
end date Ex. "2002 12 12 12 00"
output file name Ex. wl8638610.txt

Language: Bourne Shell Script

Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| dateformat | /COMF/oqcs/bin...sorc | Flexible String builder using dates. |
| mktemp | /COMF/oqcs/bin../sorc | Makes a temporary unique filename. |
| wget | /COMF/oqcs/binsgi | Web Grabber. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| $4 | Depend on requests. | Wind speed is in m/s. |

Author Name: Aijun Zhang Creation Date: 2005-01-25

**Appendix B 49 Script Name: WINDQCF.sh**

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Directory Location: /COMF/oqcs/scripts

Abstract:

Used to obtain the wind forcing files. It can read many databases and work in several modes: NWLON (PORTS stations) or NDBC (CMAN). Returns a TS2 file of the U, V surface wind field components. Produces file from tstart to tend at DT(seconds) spacing with Ueastward, Vnorthward meters/sec

y m d h m fh U V
2002 12 30 12 30 00 -5.4 3.5678

Usage: NAM station

WINDQCF.sh "-77 36.5 " NAMSTATION "2003 03 12 6 0" "2003 03 13 12 0" 0.1 NAMwind
NAM field

WINDQCF.sh "-78 -74 36 40" NAM "2002 12 15 12 30" "2002 12 16 12 30" 0.1 cbbayNAM.nc

Input Parameters:
range : field range("-78 -74 36 40", or "-77 36.5")
database : database name(NAM, or NAMSTATION)
tstart : starting time ("2005 03 22 00 00")
tend : ending time ("2005 03 28 00 00")
DT : time interval, in hours 0.10 = 6min
QCEDfile : output file name

Language: Bourne Shell Script

Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov.

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| get_data_nwlon_db.sh | /COMF/oqcs/scripts | Reads data using the SYBASE interface. |
| wind_read_ndbc.sh | /COMF/oqcs/scripts | Screen Scrapper for NDBC wind. |
| NCLwindgetNAMsub.sh | /COMF/oqcs/scripts | NCL reader of NAM file for subregions |
| NCLwindgetNAMstation.sh | /COMF/oqcs/scripts | NCL reader of ODAAS NAM file for stations. |
| cormspercent.sh | /COMF/oqcs/scripts | Calculates percentage for CORMS flags. |
| read_ndbc_archive.sh | /COMF/oqcs/scripts | Gets data from COMF/oqcs/archive/ndbc. |
| how_new.sh | /COMF/oqcs/scripts | Computes how new the forecast is. |
| how_old.sh | /COMF/oqcs/scripts | Computes how old the forecast is. |
| wind_read_nwlonweb.sh | /COMF/oqcs/scripts | Grabs wind data off the NWLON web site. |
| datemath | /COMF/oqcs/sorc | Do simple addition, subtraction of dates. |
| wind_QC_station_gapfill.f | /COMF/oqcs/sorc | Wind edit and gap filler. |
| mktemp | /COMF/oqcs/sorc | Makes a temporary unique filename. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| QCEDfile | Depend on requests. | y m d h m fh U V |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 50 Script Name: wl_read_etss.sh**

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Directory Location: /COMF/oqcs/scripts

Abstract:

Gets data from ODAAS/etss and produces an ASCII file. It must assume that the ODAAS file names will be found with: TDLDIR=$ODAASDIR/ocean/fcsts/etss/archives etssdir=`dateformat $begindate $TDLDIR/"%Y%m"/$etssname`. This does several actions to drill through the ODAAS naming conventions. First it converts the NWLON station ID to the ETSS four letter abbreviation which is used by ODAAS to identify the separate files and stations. The list of conversions is badly incomplete and needs to be filled in. Refer to this web site http://co-ops.nos.noaa.gov/active_stations.shtml.

Second, this script attempts to find a forecast file which agrees with the tstart requested. Of course it rounds down to the 00 or 12 just before $begindate. If it can't find a file by that name it will assume you want the most recent file.

The data delievered is the full forecast, so the finishing time in your request is actually ignored, and the beginning time is rounded down to the 00 or 12 hour.

Fine tuning of those times occurs in WLQCF.sh.

Usage: Interactively: wl_read_etss.sh stationid startdate enddate outputfilename
Via cron: Called by WLQCF.sh

Input Parameters: station id Ex. 8575512
start date Ex. "2002 12 10 00 00"
end date Ex. "2002 12 12 12 00"
output file name Ex. etssANNA.txt

Language: Bourne Shell Script
Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| datemath | /COMF/oqcs/bin.../sorc | Do simple addition, subtraction of dates. |
| dateformat | /COMF/oqcs/bin.../sorc | Flexible String builder using dates. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| outputfilename | Current dirctory | Has date, forecast hour, water level.<br>y m d h m fh wl |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 51 Script Name: wl_read_nwlonweb.sh**

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov
Directory Location: /COMF/oqcs/scripts
Abstract:

Grabs the water level data off the NWLON. Retrieve Preliminary (Tides) Water Level Data web page. This uses a screen scraper which directly calls the CGI used to fill in the data from http://co-ops.nos.noaa.gov/data_retrieve.shtml?input_code=101011111pwl
It depends upon this line:
echo "http://co-ops.nos.noaa.gov/cgi-bin/co-ops_qry_direct.cgi?\
stn=$stnid+1&dcp=1&ssid=A1+-+Acoustic+WL&pc=W1&datum=MLLW&\
unit=0&bdate=$bdate&edate=$edate&date=3&shift=0&level=1&form=0&\
host=&addr=10.60.7.246&data_type=pwl&format=View+Data" > $REQUESTGET
As with all screen scrapers if the CO-OPS changes this reference, then this program will crash. Output file has date, forecasthour, waterlevel
(nwlon, tide forecasthour ==0 )
y m d h m fh wl
2002 12 30 12 30 0 .5678
Now have a file with stuff like:
8638863 1 A1 1999/04/08 01:06 0.417 0.021 0 0 0 0 0
Process it into a nwlon wlnet file with:
(nwlon, tide forecasthour ==0 )
y m d h m fh wl
1999 03 01 00 00 0 0.909
1999 03 01 00 06 0 0.911
Grabs NON-Verified NWLON data is different format from verified output.txt.
Grabs verified NWLON data has to call notbracket.pl to retrieve pure data.

Usage: Interactively: wl_read_nwlonweb.sh stationid startdate enddate outputfilename
Via cron: Script is called by WLQCF.sh
Input Parameters: station id Ex. 8638610
start date Ex. "2002 12 10 00 00"
end date Ex. "2002 12 12 12 00"
output file name Ex. wl8638610.txt
Language: Bourne Shell Script
Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov
Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| dateformat | /COMF/oqcs/bin...sorc | Flexible String builder using dates. |
| mktemp | /COMF/oqcs/bin../sorc | Make a temporary unique filename. |
| wget | /COMF/oqcs/binsgi | Web grabber. |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 52 Script Name: wl_read_nwlonwebv.sh**

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Directory Location: /COMF/oqcs/scripts

Abstract: Grabs the verified water level data off the NWLON web.
It depends upon this line:
echo "http://140.90.121.76/cgi-bin/co-ops_qry_direct.cgi?\
stn=$stnid&dcp=1&ssid=WL&pc=W1+-+Six+minute&datum=MLLW\
&unit=0&bdate=$bdate&edate=$edate&date=3&shift=0&level=1&\
form=0&host=&addr=10.60.5.239&data_type=vwl&format=View+Data" >
$REQUESTGET
As with all screen scrapers if the CO-OPS changes this reference
then this program will crash.
Output file has date, forecasthour, verified waterlevel
(nwlon, tide forecasthour ==0 )
y m d h m fh vwl
2002 12 30 12 30 00 .5678

Usage: Interactively: wl_read_nwlonwebv.sh stationid startdate enddate outputfilename
Via cron: Script is called by WLQCF.sh

Input Parameters:
station id Ex. 8638610
start date Ex. "2002 12 10 00 00"
end date Ex. "2002 12 12 12 00"
output file name Ex. wl8638610.txt

Language: Bourne Shell Script
Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| dateformat | /COMF/oqcs/bin...sorc | Flexible String builder using dates. |
| mktemp | /COMF/oqcs/bin../sorc | Makes a temporary unique filename. |
| wget | /COMF/oqcs/binsgi | Web Grabber. |
| notbracket.pl | /COMF/oqcs/scripts | Perl script to help screen scrapping. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| $5 | user defined | Standard TS1 ASCII file. |

Author Name: Hong Lin Creation Date: 01-11-2005

**Appendix B 53 Script Name: WLQCF.sh**

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Hong Lin Org: NOS/CSDL
Phone: 301-713-2809x108 E-Mail: hong.lin@noaa.gov

Directory Location: /COMF/oqcs/scripts

Abstract:

Used to obtain the wind forcing files. It can read many databases and works in several modes: NWLON (PORTS stations) or NDBC (CMAN)

Returns a TS3 file of the water level, non-tidal, tidal only. (TS1, TS2 and TS3: ASCII time series with 1,2 or 3 data entries.). Produces file from tstart to tend at DT(hours) spacing fh stands for forecast hours.

y m d h min fh wl non-tidal tide-only
2002 12 29 12 30 fh .5678 .4000 .1678

Reads the observation and tide data for a station from the selected database. Then calls wlgapfill.f to do some editing and intelligent filling with the tide data. Also will ramp off of the first waterlevel observation. Parse out option to select different data base. Raw data read from ODAAS using particular data base. Returns observed water level, astronomical predicted tide and non-tidal water level.

New option "NWLONwebv" gets verified water level data from NWLON web site.

Usage: Interactively: WLQCF.sh stationid database tstart tend DT QCEDfile wlevel(tstart)
Via Cron: Called by Main_MODEL.sh

Input Parameters: sid=8638863, database=NWLON
t1="2003 02 15 12 30", t2="2003 02 16 18 36"
DT=0.10, in hours 0.10 = 6min outputfilename=CBBTWL.DAT
If wlevel(tstart) is given it will be used for ramping;
If nothing is given then no ramping will occure;
If a FILENAME is given it is assumed to be a compatible WLQCF file, the wlevel(tstart) will be read off of it and used for ramping.

Language: Bourne Shell Script

Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| get_data_nwlon_db.sh | /COMF/oqcs/scripts | Reads NWLON data using SYBASE interface. |
| cormspercent.sh | /COMF/oqcs/scripts | Calculates percentage for CORMS flags. |
| tide_read_nwlonweb.sh | /COMF/oqcs/scripts | Grabs tide data from CO-OPS web site. |
| river_read_usgs.sh | /COMF/oqcs/scripts | Screen Scrapper for USGS river discharge. |
| wl_read_etss.sh | /COMF/oqcs/scripts | Reads ETSS forecast wl from ODAAS. |
| wlgapfill.f | /COMF/oqcs/sorc | Water Level edit and gap filler. |
| datemath | /COMF/oqcs/sorc | Do simple addition, subtraction of dates. |
| mktemp | /COMF/oqcs/sorc | Makes a temporary unique filename. |
| Pred_ngofs.f | /COMF/oqcs/sorc | Gets multiple years predition of WL. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| waterlevel.out | current running directory | Text data file. |

Author Name: Tom Gross Creation Date: 2003

**Appendix B 54 Script Name: WT_read_nwlonweb.sh**

Technical Contact: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov
Aijun Zhang Org: NOS/CSDL
Phone: 301-713-2809x113 E-Mail: aijun.zhang@noaa.gov

Directory Location: /COMF/oqcs/scripts

Abstract:

Grabs the water temperature data off the NWLON. This uses a screen scraper which directly calls the CGI used to fill in the data from
http://co-ops.nos.noaa.gov/data_retrieve.shtml?input_code=101000111pan
It depends upon this line:
echo "http://www.co-ops.nos.noaa.gov/cgi-bin/co-ops_qry_direct.cgi?\
stn=$stnid&dcp=1&ssid=E1&pc=W1&datum=NULL&unit=0&bdate=$bdate\
&edate=$edate&date=3&shift=0&level=1&form=0&host=&addr=10.60.5.243\
&data_type=pan&format=View+Data" > $REQUESTGET
As with all screen scrapers if the CO-OPS changes this reference
then this program will crash.
Output file has date, forecasthour, watertemperature
(nwlon, wt forecasthour ==0 )
y m d h m fh wt
2002 12 30 12 30 0 .5678

Usage: Interactively: WT_read_nwlonweb.sh stationid startdate enddate outputfilename
Via cron: Called by TEMPQCF.sh

Input Parameters: station id Ex. 8638863
start date Ex. "2002 12 10 00 00"
end date Ex. "2002 12 12 12 00"
output file name Ex. wt8638863.txt

Language: Bourne Shell Script
Target Computer: COMF computer, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| dateformat | /COMF/oqcs/bin...sorc | Flexible String builder using dates. |
| mktemp | /COMF/oqcs/bin../sorc | Makes a temporary unique filename. |
| wget | /COMF/oqcs/binsgi | Web grabber. |

Output Files:

| Name | Directory Location | Description |
|---|---|---|
| $4 | Depend on requests. | y m d h m fh wt |

Author Name: Aijun Zhang Creation Date: 2005-01-25

## APPENDIX C. FORTRAN LIBRARY

### TABLE OF CONTENTS

Appendix C 1 catstationcurrnetcdf.f ..... 110
Appendix C 2 catstationobsnetcdf.f ..... 111
Appendix C 3 columncatfill.f ..... 112
Appendix C 4 dateformat.c ..... 113
Appendix C 5 dateformatr.c ..... 114
Appendix C 6 datemath.c ..... 115
Appendix C 7 fillnan.c ..... 116
Appendix C 8 gapfill.f ..... 117
Appendix C 9 gregorian.f ..... 118
Appendix C 10 greg2yday.c ..... 119
Appendix C 11 greg2ydaymd.c ..... 120
Appendix C 12 Hydro_netcdfs_fem.f ..... 121
Appendix C 13 Hydro_netcdfs_grid.f ..... 122
Appendix C 14 Hydro_netcdfs_station.f ..... 123
Appendix C 15 interp1.f ..... 124
Appendix C 16 julian.f ..... 125
Appendix C 17 Makefile ..... 126
Appendix C 18 mktemp.c ..... 127
Appendix C 19 obstidenetcdf.f ..... 128
Appendix C 20 pred_ngofs.f ..... 129
Appendix C 21 tripack.f ..... 130
Appendix C 22 wind_QC_station_gapfill.f ..... 131
Appendix C 23 wlgapfill.f ..... 132
Appendix C 24 wl_read_HTh.f ..... 133
Appendix C 25 wl_read_oqcs.f ..... 134

**Appendix C 1** catstationcurrnetcdf.f
Directory Location: /COMF/oqcs/sorc

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract: Reads in several files of observation data and adjusts their data onto a single time line. Assumes the data files are all the same length, keeps the -99999 values.
Then outputs a single file with
year yday u, v, u, v....
Reads input file with
year month day hour min data
Standard I/O sets up the run:
netcdffilenameout
SCRATCHDIR where all the other files exist

Usage: catstationcurrnetcdf.x < currentsobs.nc
Called by NetCDFgetstation_currents.sh

Input Parameters: Standard IO input parameters:
output file name
year month day hour starting date
year month day hour ending date
DT delta time in hours
file1
file2
file3
..
fileend

Language: lf95

Compiling/Linking Syntax :

```
F77=lf95
LIBS="-L$OQCSBIN -loqcs"
NETCDFLIB="-I$NETCDF_ROOT/include -L$NETCDF_ROOT/lib -lnetcdf"
$F77 -O catstationcurrnetcdf.f -o $OQCSBIN/catstationcurrnetcdf.x $LIBS \
$NETCDFLIB
```

Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov

Subroutines/Functions Called:

| Name | Directory Location | Description |
|---|---|---|
| Hydro_netcdfs_station.f | /COMF/oqcs/sorc/library | Writes NetCDF files for Hydro-models. |
| gregorian.f | /COMF/oqcs/sorc/library | Convert Julian days to Gregorian. |

Author Name: Tom Gross Creation Date: 2003

Remarks: Check out u, v speed's unit. Now it is divided by 1000.

**Appendix C 2 catstationobsnetcdf.f**

Directory Location: /COMF/oqcs/sorc

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract:

Reads in several files of observation data.
Assumes the data files are all the same length.
keeps the -99999 values.
Standard I/O sets up the run:
netcdffilenameout
SCRATCHDIR where all the other files exist

Reads input file with
year month day hour min sec data

Usage: catstationobsnetcdf.x < obscbbay.nc
Called by NetCDFgetstation_nwlon_fast.sh.

Input Parameters:
file.nc : A NetCDF data file.

Language: lf95

Compiling/Linking Syntax:

```
F77=lf95
LIBS="-L$OQCSBIN -loqcs"
NETCDFLIB="-I$NETCDF_ROOT/include -L$NETCDF_ROOT/lib -lnetcdf"
$F77 -O catstationobsnetcdf.f -o $OQCSBIN/catstationobsnetcdf.x $LIBS \
$NETCDFLIB
```

Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov

Suboutines/Functions Called:

| Name | Directory Location | Description |
|---|---|---|
| Hydro_netcdfs_station.f | /COMF/oqcs/sorc/library | Writes NetCDF files for Hydro-models. |
| gregorian.f | /COMF/oqcs/sorc/library | Convert Julian days to Gregorian |

Author Name: Tom Gross Creation Date: 2003

**Appendix C 3 columncatfill.f**

Directory Location: /COMF/oqcs/sorc

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract:

Column Concatenation with gap fill.
Reads in several files of observation data and adjusts their data onto a single time line.
Then outputs a single file with
year yday h1 h2 h3 h4 h5 ...
Reads input file with
year month day hour min data

Input Parameters:

fileoutput : output file name
ist_yr , ist_mon, ist_day, ist_hr : initial time
lst_yr , lst_mon, lst_day, lst_hr : last time
dt : time interval
fileinput : input filename

Language: lf95

Compiling/Linking Syntax: lf95 columncatfill.f -o columncatfill.x

Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov

Suboutines/Functions Called:

| Name | Directory Location | Description |
|---|---|---|
| interp1 | /COMF/oqcs/sorc/library | Interpolate and gap file a time series. |
| julian | /COMF/oqcs/sorc/library | Convert Gregorian dates to Julian days. |
| gregorian | /COMF/oqcs/sorc/library | Convert Julian days to Gregorian. |

Input Files:

| Name | Description |
|---|---|
| fileinput | Text file with seperated data columns |

Output Files:

| Name | Description |
|---|---|
| fileoutput | Text file with all requested columns |

Author Name: Tom Gross Creation Date: 2003

**Appendix C 4 dateformat.c**
Directory Location: /COMF/oqcs/sorc

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Astract: To create arbitrary strings using date components such as year, month, day, julian day, etc.. dateformat.c is compiled to the executable dateformat. Input year month day hour min format and it outputs date string.

Usage: dateformat year month day hour min format

```
>> dateformat 1998 3 13 13 30 "tdl%Y%m/cbbt/%Y%j%H%M.out"
>> tdl199803/cbbt/19980721330.out
```

Usage inside a script:

```
str=`dateformat 1998 3 13 13 30 "tdl%Y%m/cbbt/%Y%j%H%M.out"`
echo the string is:$str
```

It can be useful to put the dateformat, datemath routines into your path. Add this line near the beginning of your script: PATH=$PATH:/COMF/oqcs/sorc/binlinux

Notes:

So called julian year days can be used as input if they are assumed to be days of the month of January. That is Mar. 4 is yearday 63 and can be specified as either:

```
>>dateformat 1998 3  4 12 30 "%Y %m 0 %j "
>>1998 03 04 063
>>dateformat 1998 1 63 12 30 "%Y %m 0 %j "
>>1998 03 04 063
```

Time starts with Jan 01. When you want to know 90th of the day,
Using dateformatr 2005 01 90.43 0 0 "%Y %m 0 %H %M %S"
NOT dateformatr 2005 01 89.43 0 0 "%Y %m 0 %H %M %S"

General Purpose of Script Usage Example:

A directory and file system has been created with date information. We are looking for the file whose name specifies a date and time at least twelve hours before the present time but not more than 50 hours before (bailout puts the 50 hour limit. Without a bailout variable this can become an endless loop.

| Input Parameters: | | |
|---|---|---|
| | year | must be specified as 4 digits 1998, 2001 not 98 or 01 |
| | month | digits from 1 to 12 |
| | day | digits from 1 to 31 |
| | hour | digits from 1 to 24 |
| | min | digits from 0 to 59 |
| | format | a using the same syntax as the format of the UNIX command date |

Language: C
Compiling/Linking Syntax: cc dateformat.c -o dateformat.x -lm
Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov.

Author Name: Tom Gross Creation Date: Jan. 1998

**Appendix C 5 dateformatr.c**

Directory Location: /COMF/oqcs/sorc/

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract:

Date Format for Real numbers.
Operates pretty much as dateformat.
Except that the input date can have a floating point number for the day.
The fraction gets stripped off and sent into the hours, min, sec.

Useful when given a nasty floating point yday:
Evaluates real number days into hour, min, sec.
Allows use of yday conversion

Time starts with Jan 01. When you want to know 90th of the day,
Using dateformatr 2005 01 90.43 0 0 "%Y %m 0 %H %M %S"
NOT dateformatr 2005 01 89.43 0 0 "%Y %m 0 %H %M %S"

Usage: dateformatr 1999 01 250.5 0 0 "%Y%m0%H%M"
>>199909071200
dateformatr 2005 03 90.43 0 0 "%Y%m0%H%M%S"
>>20050529101912
dateformatr 2005 01 90.43 0 0 "%Y %m 0 %H %M %S"
>>2005 03 31 10 19 12

Input Parameters: year : 4 digits as 2005
month : digits from 01 to 12
day : integer or float number as 10 or 10.3
hour : integer from 1 to 24
minute : integer from 0 to 59

Language: C

Compiling/Linking Syntax: cc dateformatr.c -o dateformatr.x -lm

Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov

Author Name: Tom Gross Creation Date: 2003

**Appendix C 6 datemath.c**

Directory Location: /COMF/oqcs/sorc

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract:

Perform math on dates. Days, hours can be added or subtracted from a date. The program correctly keeps track of the roll over of hours to days, days to month etc. Differences of two full dates will give number of days and hours min separating them.
Input two dates separated by "+" or "-".
The dates are converted to Julian dates, the math worked and the result is returned as a string of year month day hour min. Usually you want to add day hour min to a full date (not two dates). No error checking is done for adding two dates.

Usage: datemath y1 mon1 d1 h1 min1 +[-] y2 mon2 d2 h2 min2

Example Usage

```
>> datemath 1998 3 2 12 30 - 0 0 3 20 0
1998 02 26 16 30
>> dateformat `datemath 1998 3 2 12 30 - 0 0 3 20 0` "%Y %h 0 %H:%M"
1998 Feb 26 16:30
```

(Note the clever use of back quotes to make that work!)

Script Usage Example:

```
str1="1998 3 2 12 30"
str2="0 0 3 20 0"
strp=`datemath $str1 - $str2`
strpdate=`dateformat $strp "%Y %h 0 %H:%M"`
echo "$str1 - $str2 = $strpdate"
```

When using "yeardays" one may think of them as days in the month of Jan.
Thus year day 180 can be written as 1998 1 180 12 30

Compilation: cc datemath.c -o datemath -lm

Input Parameters: y mon d h min are strings for the date.
+ or - is specified
Output is a string of
y mon d h m designed to be given or individual items extracted with

Language: C language.

Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov

Scripts/Programs Called:

| Name | Directory Location | Description |
|---|---|---|
| dateformat | /COMF/oqcs/binlinux | The executables. |
| dateformat.c | /COMF/oqcs/sorc | The C source code. |

Author Name: Tom Gross Creation Date: July. 1998

**Appendix C 7 fillnan.c**

Directory Location: /COMF/oqcs/sorc

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract:

Reads in a file with lines of dates, data and fills missing time delta's with NAN.

Usage:

```
cat data.dat | fillnan.x "$t1" "$t2" dt "Nan Nan" > filled_data.dat
cat data.dat | fillnan.x "2005 03 10 12 00" "2005 03 12 12 00" 0.10 "Nan Nan" >\
filled_data.dat
Ex.,
cat data.dat | ./fillnan.x "2005 03 10 13 30" "2005 03 10 15 00" 0.10 "Nan Nan" >\
filled_data.dat
cat data.dat
        2005 03 10 14 06
        2005 03 10 14 12
        2005 03 10 14 18
        2005 03 10 14 24
        2005 03 10 14 30
cat filled_data.dat
        2005 03 10 13 30 Nan Nan
        2005 03 10 13 36 Nan Nan
        2005 03 10 13 42 Nan Nan
        2005 03 10 13 48 Nan Nan
        2005 03 10 13 54 Nan Nan
        2005 03 10 14 00 Nan Nan
        2005 03 10 14 06
        2005 03 10 14 12
        2005 03 10 14 18
        2005 03 10 14 24
        2005 03 10 14 30
        2005 03 10 14 36 Nan Nan
        2005 03 10 14 42 Nan Nan
        2005 03 10 14 48 Nan Nan
        2005 03 10 14 54 Nan Nan
```

Language: C
Compiling/Linking Syntax: cc fillnan.c -o fillnan.x -lm
Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov

Author Name: Tom Gross Creation Date: 2004-01-10

**Appendix C 8 gapfill.f**

Directory Location: /COMF/oqcs/sorc

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract:

Reads in observed scalar RAWDATA file and edits and gap fills with linear ramps.
Then outputs a file (observations get artificial forecasthour)
year month day hour min fh data

Reads input file with
year month day hour min forecasthour speed dir

Usage:

```
gapfill.x << EOD > junkexec.log
$RAWDATAFILE
$OUTPUTFILE
$tstart
$tend
$DT
EOD
```

| Input Parameters: | | |
|---|---|---|
| | $RAWDATAFILE | Observation data filename. |
| | $OUTPUTFILE | Output filename. |
| | $tstart | y m d h min start of output data. |
| | $tend | y m d h min end of output data. |
| | $DT | Delta Time step in hours of output data (0.1 = six min). |

Language: Fortran 90

Compiling/Linking Syntax: lf95 gapfill.f -o gapfill.x -L/ngofs/oqcs/binlinux -loqcs

Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov

Subroutines/Functions Called:

| Name | Directory Location | Description |
|---|---|---|
| interp1 | /COMF/oqcs/sorc/library | Interpolate and gap file a time series. |
| julian | /COMF/oqcs/sorc/library | Convert Gregorian dates to Julian days. |
| gregorian | /COMF/oqcs/sorc/library | Convert Julian days to Gregorian. |

Author Name: Tom Gross Creation Date: Feb. 14, 2003

**Appendix C 9 gregorian.f**

Directory Location: /COMF/oqcs/sorc/library

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract:

Returns the year, month, day, hour given the Julian day.

Usage:

```
call GREGORIAN(jday,yr,month,day,hour)
```

Example usage:

```
real*8 julian
real*8 jday1,yr1,mon1,day1,hour1
real*8 jdayi,yri,moni,dayi,houri
read(*,*) yr1,mon1,day1,hour1
jday1=julian(yr1,mon1,day1,hour1)
dt = 1.0d/24.
do i = 1,365*24
call gregorian(jday1,yr,month,day,hour)
write(*,*) yr,month,day,hour
jday1=jday1+dt
enddo
```

Input Parameters: jday is a julian date as returned from julian()
yr,month,day,hour are the date
All variables are real*8.

Language: Fortran 77

Compiling/Linking Syntax:

From the command line with copy of the subroutine local:
f77 main.f -o main.x julian.f
Notes: Used with the julian subroutine. Note that these use the julian days defined from j=0 = 4713 B.C. Jan. 1.5
Based on equations from
Hofmann-Wellenhof B., H. Lichtenegger, and J. Collins. "Global Position System, Theory and Practice" Third revised edition, Spring-Verlag Wien New York, 1994.

Target Computer: COMF machine, ex dsofs1.nos-tcn.noaa.gov

Author Name: Tom Gross Creation Date: Jan. 1995

**Appendix C 10 greg2yday.c**

Directory Location: /COMF/oqcs/sorc/

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract:

Filters dates and data to yday and data
y m d h min data > greg2yday > y yday.frac data

Reads in a file with lines which contain the gregorian date.
Converts the gregorian dates to yday and out puts similar lines.
Sort of like a special case awk filter.

See also datemath.c and dateformat.c

Usage:

```
cat junk
1998 08 17 14 06  0.321 0.009 0 5.619  1.543 3.5 273
1998 08 17 16 06  0.187 0.012 2 5.619  1.409 6.5 281

cat junk | greg2yday
998 229.587500  0.321 0.009 0 5.619  1.543 3.5 273
1998 229.670833  0.187 0.012 2 5.619  1.409 6.5 281
```

Input Parameters: file : text file with gregorian date.

Language: C

Compiling/Linking Syntax: cc greg2yday.c -o greg2yday.x -lm

Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov

Input Files:

| Name | Directory Location | Description |
|---|---|---|
| Text file | User given | ASCII file with time data. |

Libraries Used: stdlib.h, time.h, stdio.h, string.h, math.h

Author Name: Tom Gross Creation Date: Aug 17, 1998 13:00

**Appendix C 11 greg2ydaymd.c**

Directory Location: /COMF/oqcs/sorc

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract:

Special variant on greg2yday.c.
This one only converts and replaces month day.
Keeps the hour and minutes.
y m d h min data > greg2ydaymd > y yday h min data

Reads in a file with lines which contain the gregorian date.
Converts the gregorian dates to yday and out puts similar lines.
Sort of like a special case awk filter.

See also datemath.c and dateformat.c.

Usage:

```
cat junk
1998 08 17 14 06  0.321 0.009 0 5.619  1.543 3.5 273
1998 08 17 16 06  0.187 0.012 2 5.619  1.409 6.5 281
cat junk | greg2ydaymd
1998  229 14 06 0.321 0.009 0 5.619  1.543 3.5 273
1998  229 16 06 0.187 0.012 2 5.619  1.409 6.5 281
```

Input Parameters: file: text file with gregorian date.

Language: C

Compiling/Linking Syntax: cc greg2ydaymd.c -o greg2ydaymd.x -lm

Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov

Input Files:

| Name | Directory Location | Description |
|---|---|---|
| Text file | User given | ASCII file with time data. |

Libraries Used: stdlib.h, time.h, stdio.h, string.h, math.h

Author Name: Tom Gross Creation Date: Nov 1, 2002

**Appendix C 12 Hydro_netcdfs_fem.f**

Directory Location: /COMF/oqcs/sorc/library

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract: Writes the fem NetCDF file with reals.
Includes ele for element connections, bnd for boundary listing.

Usage: call write_netcdf_Hydro_fem(netcdf_file,ncid,imode, & globalstr,ne,nn,l,nb,ele,bnd,
& time,ibasedate,lon,lat,sigma,depth,zeta,u,v,w,temp,salt,we,wn)
Optional variable writing: upon initialization set a writing variable =1.
If the variable is negative, then no variable is created or written later.
Only options are: zeta, u, v, w, temp, salt, we, wn.
Call subroutine
write_netcdf_Hydro(netcdf_file,ncid,imode,&globalstr,ne,nn,l,nb,ele,bnd,
& time,ibasedate,lon,lat,mask,sigma,depth,1.,1.,1.,0.,0.,0.,0.,0.)
This will only create and write variables zeta, u, v.

Input Parameters:

netcdf_file char*80 filename for the NetCDF output
ncid NetCDF id; generated on initialization
imode 1 for initialization, 2 for writing, 3 for closing file
globalstr Global Attributes. Set in a data statement like data globalstr/
& 'grid_type','z_type','model', 'title','comment','source',
& 'institution','history','references'/
ne dimension of the element array (number of triangles)
nn dimension of the node arrays (number of nodes)
nb dimension of the bnd array (number of boundary segments)
l dimension of vertical outputs. may be =1
ele(3,ne) Connectivity of triangular elements
bnd(4,nb) Indices of nodes making up boundary segments
bnd(1),bnd(2) nodes of a boundary segment (water on right)
bnd(3) Island number of this segment
bnd(4) Land segment=0, Water segment=1
time time in days
ibasedate(4) iyear, imonth, iday, ihour of base date (time = 0)
lon(nn) ,lat(nn) longitude, latitude of nodes
sigma(l) sigma values for vertical outputs
zeta(nn) sea surface displacement
u(nn,l),v(nn,l),w(nn,l) Velocities
temp(nn,l),salt(nn,l) Temperature, Salinity
we(nn),wn(nn) Wind Velocity Vectors (wind toward)

Language: Fortran 77
Compiling/Linking Syntax: subroutine
Target Computer: COMF machine, ex dsofs1.nos-tcn.noaa.gov
Author Name: Tom Gross Creation Date: 2003

**Appendix C 13 Hydro_netcdfs_grid.f**

Directory Location: /COMF/oqcs/sorc/library

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL

Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract: Writes the grid oriented NetCDF file with reals. NetCDF generator. IJ grid arrays. And version which scales to integers using prescribed ranges:

Usage:

```
subroutine write_netcdf_Hydro(netcdf_file,ncid,imode,
& globalstr,m,n,l,
& time,ibasedate,lon,lat,mask,sigma,depth
& ,zeta,u,v,w,temp,salt,winde,windn)
```

Optional variable writing:

Upon initialization set a writing variable =1

If the variable is negative, then no variable is created or written later.

Only options are: zeta, u, v, w, temp, salt, we, wn.

Example first call

```
subroutine write_netcdf_Hydro(netcdf_file,ncid,imode, & globalstr,m,n,l,
& time,ibasedate,lon,lat,mask,sigma,depth,1.,1.,1.,0.,0.,0.,0.,0.)
```

This will only create and write variables zeta, u, v.

Input Parameters:

```
netcdf_file   char*80  filename for the NetCDF output
ncid          NetCDF id; generated on initialization
imode  1 for initialization, 2 for writing, 3 for closing file
globalstr Global Attributes.  Set in a data statement like data globalstr/
& 'grid_type','z_type','model'
& ,'title','comment','source',
& 'institution','history','references'/
m    dimension of the X coordinate (Longitude)
n    dimension of the Y coordinate (Latitude)
l    dimension of vertical outputs. may be =1
time   time in days
ibasedate(4)  iyear, imonth, iday, ihour of base date (time = 0)
lon(m,n) ,lat(m,n)   longitude, latitude of stations
sigma(l)  sigma values for vertical outputs 0:-1 0 surface, -1 seabed
zeta(m,n)  sea surface displacement
u(m,n,l),v(m,n,l),w(m,n,l) Velocities
temp(m,n,l),salt(m,n,l)      Temperature, Salinity
we(m,n),wn(m,n) Wind Velocity Vectors (wind toward)
```

Language: Fortran 77

Compiling/Linking Syntax: subroutine

Target Computer: COMF machine, ex dsofs1.nos-tcn.noaa.gov

Subroutines/Functions Called:

| Name | Directory Location | Description |
|---|---|---|
| write_netcdf_Hydro_scale | /COMF/oqcs/sorc/library | Scales to integers by prescribed ranges. |
| check_err | /COMF/oqcs/sorc/library | Gives the error message. |

Author Name: Tom Gross Creation Date: Jan. 1995

**Appendix C 14 Hydro_netcdfs_station.f**

Directory Location: /COMF/oqcs/sorc/library

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL

Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract: Writes standardized NetCDF files for Hydro-models.
Optional variable writing: Upon initialization set a writing variable =1
If the variable is negative then no variable is created or written later written.
Only options are: zeta,u,v,w,temp,salt,wx,wy
This will only create and write variables zeta,u,v

Usage:

```
call write_netcdf_Hydro_station(netcdf_file,ncid,imode,
& globalstr,istation,stationnames,stationij,meshdim,l,
& time,ibasedate,lon,lat,sigma,depth,zeta,u,v,w,temp,salt)
call write_netcdf_Hydro_station(netcdf_file_s,ncidst,2,
& globalstr,istations,stationnames,stationij,1,nnvs,
& yday1,ibasedate,lons,lats,1.0,TOTDEPs,
& zs,Us,Vs,Ws,Ts,Ss,-1.,-1.)
```

Called by catstationobsnetcdf.f.

Input Parameters:

netcdf_file char*80 filename for the NetCDF output
ncid NetCDF id; generated on initialization
imode 1 for initialization, 2 for writing, 3 for closing file
globalstr Global Attributes.
Set in a data statement like data globalstr/
& 'grid_type','z_type','model'
& ,'title','comment','source',
& 'institution','history','references'/
istation number of output stations
stationnames char stationnames(istation)*20 Ascii station labels
stationij(istation,meshdim) indices of main mesh of the stations
meshdim dimension of the main mesh 2 for u(i,j), 1 for fem u(inode)
possibly 3 for three surrounding nodes of fem
l dimension of vertical outputs. may be =1
time time in days
ibasedate(4) iyear, imonth, iday, ihour of base date (time = 0)
lon(istation) longitude of stations
lat(istation) latitude of stations
sigma(l) sigma values for vertical outputs
zeta(istation) sea surface displacement
u(istation,l),v(istation,l),w(istation,l) Velocities
temp(istation,l) Temperature
salt(istation,l) Salinity
wx(istation), wy(istation) wind velocities (blowing toward)

Language: lf95

Target Computer: COMF machine, ex dsofs1.nos-tcn.noaa.gov

Author Name: Tom Gross Creation Date: 2003

**Appendix C 15 interp1.f**

Directory Location: /COMF/oqcs/sorc/library

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL

Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract: To interpolate a irregular spaced or gappy time series to another (probably equally spaced) time basis. Uses linear interpolation to span gaps. Uses persistence on beginning and end if original time series does not span the target times.

Example usage:

```
call interp1(nk, x, y, ni, xi, yi)
real*8 tobs(NARRAY),hobs(NARRAY)
real*8 tnew(NARRAY),hnew(NARRAY)
do i = 1,nobs
        read(10,*) tobs(i),hobs(i)
enddo
n = int((daylast-dayfirst)/dt + 1.0d00)
do i = 1 , n
        tnew(i) = dayfirst+(i-1)*dt
enddo
call interp1(nobs,tobs,hobs,n,tnew,hnew)
```

The resultant array hnew(tnew) is an interpolated function agreeing with hobs(tobs).

Input Parameters:

x,y are filled data vectors
xi is vector of new xi,yi
yi will be filled with interpolated values at xi inside x,y
x should be monotonic
It won't bomb otherwise, but don't trust the method!
If xi is less than min(x) pre-persistence fills in the blanks
If xi is greater than max(x) persistence fills in the blanks

Language: Fortran 77

Compiling/Linking Syntax:

From the command line with copy of the subroutine local:

```
f77 main.f -o main.x interp1.f
```

Using a makefile and the mmapf library for the SGI OPSEA:

This makefile may be used to create an executable for main.f

```
FF = f90 -extend_source -n32 -r4 -mips4 -O2 -static
my_executable_name = main.x
mysource.f = main.f
mysubs.f =
libsrc = /opseadisk3/MMAPlib/libmmapf.a
lib = -L/opseadisk3/tgross/MMAPENVIRON/progs/lib -lmmapf
$(my_executable_name): $(mysource.f) $(mysubs.f) $(libsrc)
$(FF) -o $(my_executable_name) $(mysource.f) $(mysubs.f) $(lib)
```

Target Computer: COMF machine, ex dsofs1.nos-tcn.noaa.gov

Author Name: Tom Gross Creation Date: Jan. 1998

**Appendix C 16 julian.f**
Directory Location: /COMF/oqcs/sorc/library

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract: Returns double precision Julian day given a Gregorian date (year, month,day,hour).

Usage: jday= JULIAN( year,month,day,hour)

Example usage:

```
real*8 julian
real*8 jday1,yr1,mon1,day1,hour1
real*8 jdayi,yri,moni,dayi,houri
read(*,*) yr1,mon1,day1,hour1
jday1=julian(yr1,mon1,day1,hour1)
dt = 1.0d/24.
do i = 1,365*24
        call gregorian(jday1,yr,month,day,hour)
        write(*,*) yr,month,day,hour
        jday1=jday1+dt
enddo
```

Input Parameters: yr, month, day, hour are the date.
jday is julian day.
julian days defined from j=0 = 4713 B.C. Jan. 1.5
All variables are real*8.
Because this is a real*8 function, the function itself must be declared with real*8 julian.

Language: Fortran 77
Compiling/Linking Syntax: From the command line with copy of the subroutine local:
f77 main.f -o main.x julian.f
Notes: Used with the gregorian subroutine which is also contained in julian.f file.
Note that these use the julian days defined
from j=0 = 4713 B.C. Jan. 1.5
year days are easily calculated by doing:
yday = julian(year,month,day,hour) - julian(year,1,1,0) +1.0doo
Based on equations from
Hofmann-Wellenhof B., H. Lichtenegger, and J. Collins.
"Global Position System, Theory and Practice" Third revised edition, Spring-Verlag Wien New York, 1994.
Target Computer: COMF machine, ex dsofs1.nos-tcn.noaa.gov

Author Name: Tom Gross Creation Date: Jan. 1995

**Appendix C 17 Makefile**

```
FC = lf95
F77 = lf95
LF77 = lf95
# lf95 recommended run flags
FFLAGS = --nap --nchk --ng -O --npca --nsav --ntrace \
        --wide --ml cdecl -I/usr/local/include
ARFLAGS = rv
RM = rm
LIB = libquodinit.a
LIBS = -L/usr/local/lib -L/COMF/oqcs/binlinux -loqcs -lnetcd

SRCS = quoddy5_1.1_coresubs.f quoddy5_1.1_usrsubs_resources.f \
q511NMLPAKS_000607.f DCMSPAK_010905.f                         \
tideadcirc.f usrsubs1999.f tideadcircsubs.f                   \
filets.f pointsource5.f atmosfourfilesnow.f rampupcold.f      \
>write_netcdf.f

OBJS = ${SRCS:.f=.o}

main:       libquodinit.a quoddy5_1.1_main.f
      $(LF77) $(FFLAGS) quoddy5_1.1_main.f $(LIB) \
                    -o q511init.x $(LIBS)


libquodinit.a:   ${OBJS}
   ar ${ARFLAGS} $@ $?
```

**Appendix C 18 mktemp.c**

Directory Location: /COMF/oqcs/sorc

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract:

Makes a temporary unique filename from a template ROOTNAME.XXXXXX. The XXXXXX is replaced by a unique string.

SGI , as a crippled version of UNIX, does not provide the external wrapper for the C subroutine mktemp. This just dummies up only the functionality of the -q option as demonstrated above. Don't expect anything else (unless you switch to LINUX).

Usage: TMPFILE=`mktemp -q ROOTNAME.XXXXXX`

Compilation:

cc mktemp.c -o mktemp -lm

Input Parameters: Rootname: the file's root name.

Language: C language.

Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov

Author Name: Tom Gross Creation Date: Feb. 13, 2003

Remarks:

A small warning: When compiled on LINUX this routine will actually create and open the file, rather than only return the string. So don't use it on Linux.

**Appendix C 19 obstidenetcdf.f**

Directory Location: /COMF/oqcs/sorc

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL

Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract: Combines several single time series files into a single NetCDF "Station" file. Used to create the obs.nc and tides.nc files which accompany the modelnowstation.nc and modelforestation.nc files which are used for graphics generation.

The program keeps reading and processing this info till it runs out.

The NetCDF files then contain gap filled data on the new time line spacing.

Usage: Requires quite a few input files and control info read from standard IO.

The two output files are always named tide.nc and obs.nc

Construct the input file with the dates and netcdfoutputfilename.

```
cat << EOD > $SCRATCHDIR/columncat1.input
SCRATCHDIRgetnwlon
`dateformat $begindate "%Y %m 0 %H "`
`dateformat $enddate "%Y %m 0 %H "`
.1
EOD
cat $SCRATCHDIR/columncat1.input $stationdata > $SCRATCHDIR/columncat.input
obstidenetcdf.x < $SCRATCHDIR/columncat.input > /dev/null
mv obs.nc $netcdfout"_obs.nc"
mv tide.nc $netcdfout"_tide.nc"
```

Called by NetCDFgetstations_astro.sh

Input Parameters:

| | |
|---|---|
| scratchdirectoryname | Directory where input files are found |
| starting date | % Time series are built spanning these |
| ending date | |
| Delta time | % Time step of output time series |

8638863 cbbt "Chesapeake Bay Bridge Tunnel"
36 58.0 N 76 6.8 W
8574680 balt "Baltimore"
39 16.0 N 76 34.7 W
8573364 tolc "Tolchester"
39 12.8 N 76 14.7 W

Language: lf95

Compiling/Linking Syntax: lf95 obstidenetcdf.f -o obstidenetcdf.x \
-I/usr/local/include -L/usr/local/lib -L/ngofs/oqcs/binlinux -loqcs -lnetcdf

Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov

Suboutines/Functions Called:

| Name | Directory Location | Description |
|---|---|---|
| Hydro_netcdfs_station.f | /COMF/oqcs/sorc/library | Writes NetCDF files for Hydro-models. |
| interp1.f | /COMF/oqcs/sorc/library | Interpolate and gap file a time series. |
| gregorian.f | /COMF/oqcs/sorc/library | Convert Julian days to Gregorian. |
| julian.f | /COMF/oqcs/sorc/library | Convert Gregorian dates to Julian days. |

Author Name: Tom Gross Creation Date: 2003

**Appendix C 20 pred_ngofs.f**

Directory Location: /COMF/oqcs/sorc

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL

Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract: This program is modified from pred.f of Chris Zervas so that it can make prediction of multiple years. Change call CONCTJ and CONJTC to call julian. Also call equarg.f to calculate XODE and VPU, instead of reading from data file 'yr'.

Usage:

pred_ngofs.x "$begindate" "$enddate" $kindat $DELT $CONV $XMAJOR $filein $fileout

Called by NetCDFgetstation_currents.sh.

Input Parameters:

```
BEGINDATE="2005 01 01 12 30"
ENDDATE= "2005 12 31 12 30"
KINDAT=1, for current prediction; =2 for water level prediction
DELT is time interval of output time series in hours
CONV: Units convertion of predicted variable
XMAJOR is principle current direction in degrees
filein is input file name which includes tide constituents
fileout is output name which contains predicted water level or current time series
1   2   1.  0.   0          ! nsta ipredk conv tconv il2
tss.out                      ! Output time series file
0  4  15 0  6 30 0 0.1 1998 1998 106.0
IEL,IMMS,IDDS,TIME,IMME,IDDE,TIMEL,DELT,IYRS,IYRE,XMAJOR
        Harmonic Analysis of Data in  325j4b05.dat
        29-Day H.A.  Beginning  4-15-1998  at Hour 17.30  along 106 degrees
        12718
19316218281400221151866416411179733 76 68733224 81743495 5868 163
2   0  0  804  92   0   0 36211666  4431590   0   0 24821455
3  3513257  6521961   0   0  5803436  6463317   0   0   0   0
0   0   0   0   0   0  3113546 15863554  8262104  1122127
5  213  13 39053385   0   0   0   0 26691641   0   0 38082138
6 28911704   0   0
Harmonic Analysis of Data in  325j4b05.dat
29-Day H.A.  Beginning  4-15-1998  at Hour 17.30  along 196 degrees
        -5820
1 57222694 14073452 22192976 1203 154 47261638 1292 478 26243445
2   0  0  658 796   0   0  4302938  6232349   0   0  2953258
3   563430   403046   0   0   92 316  1023593   0   0   0   0
4   0   0   0   0   0   0   49 617  251 639   833422   113482
5   34 800  398 178   0   0   0   0  3172976   0   0  3833513
6 12661394   0   0C
```

Language: lf95

Compiling/Linking Syntax: lf95 pred_ngofs.f -o pred_ngofs.x

Target Computer: COMF machine, ex dsofs1.nos-tcn.noaa.gov

Author Name: Chris Zervas

**Appendix C 21 tripack.f**

Directory Location: /COMF/oqcs/sorc

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL

Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract:

A collection of subroutines which can be used to create a triangulated mesh from a set of randomly distributed X,Y points. Used by some of the models to map a distributed set (or a regular array) of wind values to the model grid.

Input Parameters:

NCC = Number of constraint curves (constraint regions). NCC .GE. 0.

LCC = Array of length NCC (or dummy array of length 1 if NCC = 0) containing the index (for X, Y, and LEND) of the first node of constraint I in LCC(I) for I = 1 to NCC. Thus, constraint I contains K = LCC(I+1) - LCC(I) nodes, K .GE. 3, stored in (X,Y) locations LCC(I), ..., LCC(I+1)-1, where LCC(NCC+1) = N+1.

N = Number of nodes in the triangulation, including constraint nodes. N .GE. 3.

X,Y = Arrays of length N containing the coordinates of the nodes with non-constraint nodes in the first LCC(1)-1 locations, followed by NCC sequences of constraint nodes. Only one of these sequences may be specified in clockwise order to represent an exterior constraint curve (a constraint region with nonfinite area).

The above parameters are not altered by this routine.

LWK = Length of IWK. This must be at least 2*NI where NI is the maximum number of arcs which intersect a constraint arc to be added. NI is bounded by N-3.

IWK = Integer work array of length LWK (used by EDGE to add constraint arcs).

LIST,LPTR,LEND = Data structure defining the triangulation. Refer to TRMESH.

On output:

LWK = Required length of IWK unless IER = 1 or IER =3. In the case of IER = 1, LWK is not altered from its input value.

IWK = Array containing the endpoint indexes of the new arcs which were swapped in by the last call to Subroutine EDGE.

LIST,LPTR,LEND = Triangulation data structure with all constraint arcs present unless IER .NE. 0. These arrays are not altered if IER = 1.

IER = Error indicator:

IER = 0 if no errors were encountered.

IER = 1 if NCC, N, or an LCC entry is outside its valid range, or LWK .LT. 0 on input.

IER = 2 if more space is required in IWK.

IER = 3 if the triangulation data structure is invalid, or failure (in EDGE or OPTIM) was caused by collinear nodes on the convex hull boundary. An error message is written to logical unit 6 in this case.

IER = 4 if intersecting constraint arcs were encountered.

IER = 5 if a constraint region contains a node.

Language: Fortran

Compiling/Linking Syntax: subroutine

Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov

Author Name: Robert J. Renka Creation Date: 11/12/94

**Appendix C 22 wind_QC_station_gapfill.f**

Directory Location: /COMF/oqcs/sorc

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract:

Reads in observed wind speed and direction RAWDATA file, and edits and gap fills with linear ramps.
Then outputs a file
year month day hour min Ueastward Vnorthward
Reads input file with
year month day hour min forecasthour speed dir

Usage: wind_QC_station_gapfill.x $RAWDATAFILE $OUTPUTFILE $tstart \
$tend $DT > junkexec.log

Input Parameters:

| | |
|---|---|
| $RAWDATAFILE | Observation wind data filename |
| $OUTPUTFILE | Output filename |
| $tstart | y m d h min start of output data |
| $tend | y m d h min end of output data |
| $DT | Delta Time step in hours of output data (0.1 = six min) |

Language: Fortran 90

Compiling/Linking Syntax:

```
lf95 wind_QC_station_gapfill.f -o wind_QC_station_gapfill.x –L ../binlinux -loqcs
```

Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov

Suboutines/Functions Called:

| Name | Directory Location | Description |
|---|---|---|
| interp1 | /COMF/oqcs/sorc/library | Interpolate and gap file a time series. |
| julian | /COMF/oqcs/sorc/library | Convert Gregorian dates to Julian days. |
| gregorian | /COMF/oqcs/sorc/library | Convert Julian days to Gregorian. |

Author Name: Tom Gross Creation Date: Jan. 31, 2003

**Appendix C 23 wlgapfill.f**

Directory Location: /COMF/oqcs/sorc

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL

Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract:

Called by WLQCF.sh

Reads in the observed raw waterlevel observation and tide files. Edits, and gap fills intelligently with linearly interpolated non-tidal plus astro tides. Then outputs a file year month day hour min 0 wlobs wlobs-non-tide wltide.

The 0 is a standin for forecast hour.

Produces the TS3 files with date, obs, non-tide, astro-tide.

An option allows the input observed data file to be the Non-Tidal data and a "obs" time series is created using that plus the astro-tide. The starting time observation value may be specified along with a ramp length. This can be used to smoothly join the new data series with a previously created data series which ended on the new file's start time with the value of wlstart. Reads input file with year month day hour min forecasthour data.

Usage:

```
wlgapfill.x << EOD > junkexec.log
 $WLDAT
 $TIDEDAT
 $QCEDfilename
 $wltype
 $tstart
 $tend
 $DT
 $wlstart
 $ramphours
 EOD
```

Input Parameters:

| | |
|---|---|
| $WLDAT | Observation water level data filename |
| $TIDEDAT | Astro Tide Only data filename |
| $QCEDfilename | Output filename |
| $wltype | TIDAL or NON-TIDAL , type of obs data |
| $tstart | y m d h min start of output data |
| $tend | y m d h min end of output data |
| $DT | Delta Time step in hours of output data (0.1 = six min) |
| $wlstart | Force water level at $tstart to equal this |
| $ramphours | Ramp from wlstart to normal over ramphours |

Language: Fortran 90

Compiling/Linking Syntax: lf95 wlgapfill.f -o wlgapfill.x

Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov

Suboutines/Functions Called:

| Name | Directory Location | Description |
|---|---|---|
| interp1 | /COMF/oqcs/sorc/library | Interpolate and gap file a time series. |
| julian | /COMF/oqcs/sorc/library | Convert Gregorian dates to Julian days. |

Author Name: Tom Gross Creation Date: Dec. 26, 2002

**Appendix C 24 wl_read_HTh.f**

Directory Location: /COMF/oqcs/sorc/library

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract:

Partially to library subroutines.
Also to extract the H(Th) from a wl file.
Called by WLQCF.sh.

Usage: wl_read_HTh.x $inoutfilename $timestart

Input Parameters: $inoutfilename : an input file name
$timestart : starting time.

Language: Fortran 95

Compiling/Linking Syntax:
lf95 wl_read_HTh.f wl_read_oqcs.f julian.f interp1.f -o ../../binlinux/wl_read_HTh.x

Target Computer: Runs on COMF computers, such as dsofs1.nos-tcn.noaa.gov

Subroutines/Functions Called:

| Name | Directory Location | Description |
|---|---|---|
| wl_read_oqcs.f | /COMF/oqcs/sorc/library | Reads a water level file. |
| interp1 | /COMF/oqcs/sorc/library | Interpolate and gap file a time series. |
| julian | /COMF/oqcs/sorc/library | Convert Gregorian dates to Julian days. |

Author Name: Tom Gross Creation Date: 2003

**Appendix C 25 wl_read_oqcs.f**

Directory Location: /COMF/oqcs/sorc/library

Technical Contact(s): Name: Tom Gross Org: NOS/CSDL
Phone: 301-713-2809x139 E-Mail: tom.gross@noaa.gov

Abstract:

READS a water level file with
year, month,day, hour, min, fhour, obs, nontidal, tidal

Usage: call wl_read_oqcs(fileinput,NN,year,month,day,hour,minute,
& fhour,obs, nontidal, tidal )

Input Parameters: fileinput : input filename
NN : counter
year,month,day,hour,minute : integer time
fhour : forecast hour
obs, nontidal, tidal : observation, tidal, non-tidal data.

Language: Fortran 77

Compiling/Linking Syntax: subroutine

Target Computer: COMF machine, ex dsofs1.nos-tcn.noaa.gov

Author Name: Tom Gross Creation Date: 2003

## APPENDIX D. SAMPLE MODEL MAIN SCRIPT (CBOFS)

```
#!/bin/sh
# MAIN_CBOFS.sh
# Runs a nowcast and forecast with Mecca using a hotstart file
#
# Put all the directories into environment variables
# copy of setenvironmentvariables.sh
#
# ##########################################
# gbofs1.nos.noaa.gov  /COMFf/ohms/cbofs
# ##########################################


# Flag set which will turn on the graphics
# Default or no flag, or anything other than "DO_GRAPHICS"
# will suppress graphics
if [ $# -eq 1 ]
then
DO_GRAPHICS=$1
else
DO_GRAPHICS=DO_NOT_DO_GRAPHICS
fi


##############################################################################
#   MODULE 0 Set Environment Variables for Directories
##############################################################################

# These are now set in the crontab file
#source /comf/staging/COMF/oqcs/setenvironmentvariables_staging.sh
#export MODELDIR=/comf/staging/COMF/ohms/CBOFS

echo "First line of main-cbofs.sh   "$MODELDIR

export MODELWORK=$MODELDIR/work
export MODELBIN=$MODELDIR/binlinux
export ARCHIVEDIR=$MODELDIR/archive
export MODELWWW=$MODELDIR/wwwgraphics
export MODELINFO=$MODELDIR/info

export PATH=$PATH:$MODELBIN
export MODELLOGDIR=$MODELDIR/execlog
export CORMSLOG=$MODELWORK/corms_raw.txt
```

```
##############################################################################
##############################################################################
# Change to Model WORK directory
# All intermediate and scratch files land here
cd $MODELWORK
##############################################################################
##############################################################################

echo "  MAIN_CBOFS.sh Started at RealTimeClock          " `date`
echo "  MAIN_CBOFS.sh running from "$MODELWORK


##############################################################################
#  MODULE 1 OFS System Tests
##############################################################################
if [ $COMFDIR ]
then
echo " The operational system directories are environment variables:"
echo  "COMFDIR    " $COMFDIR
echo  "OQCSDIR   " $OQCSDIR
echo  "OQCSBIN   " $OQCSBIN
echo  "ODAASDIR   " $ODAASDIR
echo  "OPDSDIR    " $OPDSDIR
else
echo " The operational system directories are environment variables"
echo " Set with:    source /COMF/oqcs/setenvronmentvariables.sh"
echo " They are not set.  Abort this run "
exit
fi
if [ -e $MODELWORK ]
then
echo " The Model directory exists and:"
echo  "MODELWORK     " $MODELWORK
echo  "MODELBIN      " $MODELBIN
echo  "ARCHIVEDIR    " $ARCHIVEDIR
echo  "MODELWWW      " $MODELWWW
echo  "MODELINFO=    " $MODELINFO
else
echo " The MODELWORK directory doesn't exist"
echo " They are not set.  Abort this run "
exit
fi

# Calling control script to check for prior instances of model script process.
# If prior instances exist or $MODELDIR/info/ofs_control_prevented exists,
# MAIN_CBOFS.sh script will be terminated.
```

OFS_CONTROL.sh

```
echo "MAIN_CBOFS.sh Started at" `date`  > $CORMSLOG

df | grep archive | tr % .    |awk '{print "DISKFREE ARCHIVE " 100-$5}' \
  | head -n 1 >> $CORMSLOG
df | grep odaas1 | tr % .    |awk '{print "DISKFREE ODAAS  " 100-$5}' \
          >> $CORMSLOG

#########################################

# clean out previous runs for CORMS testing
rm gentide_now.out gen2obs.out1a


##############################################################################
#   MODULE 2  Create the start and stop times
##############################################################################

# Nowcast starts with an existing HOTSTART.DAT
cp $MODELDIR/init/HOTSTART.DAT $MODELWORK/.
cp $MODELDIR/init/wlcbbtHOTSTART.dat $MODELDIR/work/.
# Start with time read from the hotstart file
time_hotstart=`readinitspace.x << EOD
"HOTSTART.DAT"
EOD`

# run up to NOW
time_now=`date -u +"%Y %m %d %H 0"`
time_nowcastend=$time_now
echo $time_nowcastend > $MODELINFO/timetest.dat
time_forecastend=`datemath $time_now + 0 0 0 24 0`
# time for forcing data files to end.  Need the 17 min offset gap
time_nowcastend=`date -u +"%Y %m %d %H 30"`
echo $time_nowcastend >> $MODELINFO/timetest.dat

echo "CBOFS NOWCAST "$time_hotstart" "$time_now >> $CORMSLOG
echo "Run from $time_hotstart  to  $time_now "



#########################################
#   Module 3 Get data for  NOWCAST
#########################################
```

```
echo " Get CBBT water level for outer forcing"
# use the last wlcbbt.dat file to assure no jumps in water levelforcing
WLQCF.sh 8638863 NWLON "$time_hotstart" "$time_nowcastend" 0.10 wlcbbt.dat
wlcbbtHOTSTART.dat > $MODELLOGDIR/WLQCF.log


echo "Get CBBT wind and TPLM2 wind for forcing"

WINDQCF.sh "TPLM2" NDBC "$time_hotstart" "$time_nowcastend" 0.10 windtplm.dat >
$MODELLOGDIR/WINDQCF1.log
WINDQCF.sh 8638863 NWLON "$time_hotstart" "$time_nowcastend" 0.10 windcbbt.dat >
$MODELLOGDIR/WINDQCF2.log

echo "Get Rivers  Climatological"
cp $MODELINFO/rivers.met $MODELWORK/.


##########################################
#   Module 4 Reformat data for NOWCAST
##########################################

# translate wlcbbt.dat  mllw,     obs, non-tidal, tidal
#2003  1 22  0  0  0              0.2140    0.0270    0.1870
#2003  1 22  0  6  0              0.2320    0.0240    0.2080
# scale tidal component with amp, phase shift
# Phase shift of -0.0118056 days = -17min
# then convert non-tidal  mllw to msl  -.442,
# finally add back the amplitude corrected tide 1.134
# alternative methods  ($8-.442)+($9*1.134) or 1.134*($8+$9-.441)
# and format year, yearday,  wl
# data/gentide_now.out
# 2003  21.988194  -0.202942
# 2003  21.992361  -0.182128
#
awk '{ print $1 " " $2 " " $3" " $4  " " $5-17 " " ($8+($9-.442)*1.134) }' \
  wlcbbt.dat  | greg2yday.x > gentide_now.out

if test -s gentide_now.out ; then CORMSPERCENT=100 ; else CORMSPERCENT=0 ; fi
echo "GENTIDE NOW "$CORMSPERCENT >> $CORMSLOG


# use genwind_2obsoqcs to convert windtplm.dat windcbbt.dat to data/genwind_NOW.out
# NOWcbbt.met, NOWtplm.met  > genwind_2obsB.x  > genwind_now.out
# Needs to use ydays
cat windcbbt.dat | greg2yday.x >NOWcbbt.met
cat windtplm.dat | greg2yday.x >NOWtplm.met
```

```
#########
# Build genwind.input and execute genwind_2obsoqcs..x
#########
echo "Start genwind_2obsoqcs.x"
# And make sure its long enough by ending 4 hours later (3+3/24 =.25)
diffdate=`datemath $time_nowcastend - $time_hotstart `
Zlengthdays=`echo $diffdate | awk '{ print $3 + $4/24 + 0.25 }'`

genwind_2obsoqcs.x <<EOD > $MODELLOGDIR/genwindnow.log
`dateformat $time_hotstart "%Y"`
`dateformat $time_hotstart "%j"`
`dateformat $time_hotstart "%H"`
$Zlengthdays
NOWcbbt.met
NOWtplm.met
EOD


if test -s gen2obs.out1a ; then CORMSPERCENT=100 ; else CORMSPERCENT=0 ; fi
echo "GENWIND NOW "$CORMSPERCENT >> $CORMSLOG

mv gen2obs.out1a $MODELWORK/genwind_now.out



# # # #
```

```
sed -e s/VHOURSV/$hourslength/ \
   -e s/VIYEAR0V/$ZY/ \
   -e s/VIMONTH0V/$ZM/ \
   -e s/VIDAY0V/$ZD/ \
   -e s/VIHOUR0V/$ZH/ \
   $MODELDIR/info/templates/now.con.template > now.con

cp $MODELDIR/info/fullbay21c.geo $MODELWORK/.
cp $MODELDIR/info/fullbay21c.ll $MODELWORK/.

echo "Start Mecca NOWCAST RUN"
echo "$MODELBIN/mecca21nclf95.x < now.con > $MODELLOGDIR/pr.now"

which mecca21nclf95.x

mecca21nclf95.x < now.con > $MODELLOGDIR/pr.now

echo " Done with Mecca nowcast run"
tail -20 $MODELLOGDIR/pr.now

# Check pr.now for successful run
istop=`tail -1 $MODELLOGDIR/pr.now`
echo "******DEBUGING OF istop  istop="$istop"="
if [ "$istop" = " ISTOP= 0" ]; then CORMSPERCENT=100 ; else CORMSPERCENT=0 ; fi
echo "NOWCAST DONE "$CORMSPERCENT >> $CORMSLOG
echo "NOWCAST DONE  "$CORMSPERCENT

# Move the results
# Move the HOTSTART.DAT and its waterlevel forcing file
# These will be used by the Forecast
mv  nowFIN.DAT $MODELDIR"/init/nowFIN.DAT"
cp  wlcbbt.dat $MODELDIR"/init/nowFINwlcbbtHOTSTART.dat"

# Prepare netcdf files for copying by ARCHIVE.sh  (standard names)
cp meccastation.nc stationsnow.nc
cp mecca2d.nc fieldsnow.nc
cp $MODELDIR"/init/nowFIN.DAT" hotstartout
cp $MODELDIR"/init/nowFINwlcbbtHOTSTART.dat" hotstartout.wlcbbt
tar -cvf modelinput.tar  now.con fullbay21c.geo genwind_now.out gentide_now.out rivers.met

echo "  Nowcast Finished at RealTimeClock          " `date`


##########################################
#  MODULE 2  Create the start and stop times for the FORECAST
```

```
#########################################
# Nowcast ends with this HotStart file:
cp  $MODELDIR/init/nowFIN.DAT $MODELDIR/work/HOTSTART.DAT
cp $MODELDIR"/init/nowFINwlcbbtHOTSTART.dat"  $MODELWORK/wlcbbtnow.dat
time_nowcastend=`$MODELBIN/readinitspace.x << EOD
"HOTSTART.DAT"
EOD`
time_forecastend=`datemath $time_nowcastend + 0 0 0 24 0`
time_hotstart=$time_nowcastend

echo "FORECAST "$time_hotstart" to "$time_forecastend
#########################################
#   Module 3 Get data for  FORECAST
#########################################

#############################################
#  Forecast water level  ETSS
#############################################
WLQCF.sh 8638863 ETSS "$time_nowcastend" "$time_forecastend" 0.10 \
   wlcbbtfore.dat wlcbbtnow.dat > $MODELLOGDIR/WLQCF.log

#############################################
echo "Get Rivers  Climatological"
cp $MODELDIR/info/rivers.met $MODELWORK/.

#############################################
#  Forecast wind field, Get NAM file.nc
rm windsNAM.nc windsNAM.bin windsmecca.bin
WINDQCF.sh "-78 -74 36 40" NAM "$time_nowcastend" "$time_forecastend" 1.0 \
    windsNAM.nc > windNAMqcf.log


#########################################
#   Module 4 Reformat data for FORECAST
#########################################

#############################################
#  Forecast water level  ETSS
#############################################
# translate wlcbbt.dat  mllw,     obs, non-tidal, tidal
#2003  1 22  0  0  0            0.2140   0.0270   0.1870
#2003  1 22  0  6  0            0.2320   0.0240   0.2080
# scale tidal component with amp, phase shift
# Phase shift of -0.0118056 days = -17min
# then convert non-tidal  mllw to msl  -.442,
# finally add back the amplitude corrected tide 1.134
```

```
# alternative methods  ($8-.442)+($9*1.134) or 1.134*($8+$9-.441)
# and format year. yearday.  wl
# data/gentide_now.out
# 2003  21.988194  -0.202942
# 2003  21.992361  -0.182128
awk '{ print $1 " " $2 " " $3" " $4  " " $5-17 " " ($8+($9-.442)*1.134) }' \
  wlcbbtfore.dat | greg2yday.x > gentide_fore.out

if test -s gentide_fore.out ; then CORMSPERCENT=100 ; else CORMSPERCENT=0 ; fi
echo "GENTIDE FORE "$CORMSPERCENT >> $CORMSLOG

##############################################
#  Forecast wind field, Get NAM / NAM.nc, Convert to mecca.bin
# genwind_subs_net.x also scales by 1.2 and does Lambert Rotation
##############################################

genwind_subs_net.x << EOD
$time_nowcastend
48
2
windsNAM.nc
windsNAM.bin
windsmecca.bin
EOD

if test -s windsmecca.bin ; then CORMSPERCENT=100 ; else CORMSPERCENT=0 ; fi
echo "GENWIND FORE "$CORMSPERCENT >> $CORMSLOG

mv windsmecca.bin $MODELWORK/genwind_fore.out


# # # #
```

```
datediff=`datemath $time_forecastend - $time_nowcastend `
hourslength=`echo $datediff | awk '{ print $3*24+$4+$5/60}'`
ZY=`dateformat $time_nowcastend "%Y"`
ZM=`dateformat $time_nowcastend "%m"`
ZD=`dateformat $time_nowcastend "%d"`
ZH=`dateformat $time_nowcastend "%H"`
sed -e s/VHOURSV/$hourslength/ \
   -e s/VIYEAR0V/$ZY/ \
   -e s/VIMONTH0V/$ZM/ \
   -e s/VIDAY0V/$ZD/ \
   -e s/VIHOUR0V/$ZH/ \
   $MODELDIR/info/templates/fore.con.template > fore.con

cp $MODELDIR/info/fullbay21c.geo $MODELWORK/.


echo "Start Mecca 24hr FORECAST RUN"
echo "$MODELBIN/mecca21nclf95.x < fore.con > $MODELLOGDIR/pr.fore"

mecca21nclf95.x < fore.con > $MODELLOGDIR/pr.fore

echo " Done with Mecca forecast run"
tail -20 $MODELLOGDIR/pr.fore

# Check pr.now for successful run
istop=`tail -1 $MODELLOGDIR/pr.fore`
if [ "$istop" = " ISTOP= 0" ]; then CORMSPERCENT=100 ; else CORMSPERCENT=0 ; fi
echo "FORECAST DONE "$CORMSPERCENT >> $CORMSLOG

# Move the results
# Prepare netcdf files for copying by ARVHIVE.sh  (standard names)
cp meccastation.nc stationsfore.nc
cp mecca2d.nc fieldsfore.nc
tar -rvf modelinput.tar gentide_fore.out genwind_fore.out fore.con

echo "  CBOFS FORECAST Finished at RealTimeClock        " `date`



#########################################
#  Module 6 Archive the data
#########################################
export ARCHIVEDIR=$MODELDIR/archive
time_roundhour=`datemath $time_nowcastend + 0 0 0 0 10`
time_roundhour=`dateformat $time_roundhour "%Y %m %d %H 00"`
echo "ARCHIVE.sh CBOFS \"$time_roundhour\" \"$time_roundhour\" "
```

```
ARCHIVE.sh CBOFS "$time_roundhour" "$time_roundhour"


if [ $DO_GRAPHICS = "DO_GRAPHICS" ]
then
#########################################
#  Module 7 Make the graphics (All files are pulled from ARCHIVEDIR)
#########################################
# required! Before and after GRAPHICS temporarily rename and
#         export CORMSLOG to keep WLQCF flags from clobbering.
OLDCORMSLOG=$CORMSLOG
export CORMSLOG=$MODELWORK/cormsscratchgraphics.txt

rm $MODELWORK/*.png $MODELWORK/*.ctl
cp $MODELDIR/info/plot_timeseries_wl_cbofs.ctl plot_timeseries_wl.ctl
cp $MODELDIR/info/plot_field_cbofs2.ctl plot_field.ctl
TIME_NOWCASTSTART=`datemath $time_roundhour - 0 0 0 24 0 `

# cbofs.ctl uses /ngofs/ohms/cbofs2/work/stationsnow.nc  stationsfore.nc
# With the INIT timing these files are always long enough
# so no need to concatenate nowcasts with:
#grabarchivenetcdf.sh "$TIME_NOWCASTSTART" "$time_forecastend"\
#    _CBOFS_stationsnow.nc stationsnow.nc

echo "Start Graphics.sh "$TIME_NOWCASTSTART" -> "$time_forecastend
pwd


datedir=`dateformat $time_nowcastend
"$MODELDIR/execlog/%Y%m%d%H00_diagnostics_graphics.log"`

echo GRAPHICS.sh  \"$TIME_NOWCASTSTART\" \"$time_forecastend\"  ">" $datedir

GRAPHICS.sh  "$TIME_NOWCASTSTART" "$time_forecastend"  >  $datedir



#################################
# required
export CORMSLOG=$OLDCORMSLOG
# produce a corms flag based on number of graphics
# the number of graphics  is approx. 144 normally
numpngs=`ls -1 $MODELWORK/*.png | wc -l`
  echo "GRAPHICS DONE $numpngs" >> $CORMSLOG
```

```
# Remove all files from wwwgraphics
rm -fr $MODELWWW/*

ARCHIVE_GRAPHICS.sh "$time_nowcastend"
echo "AFTER MODULE  MODULE 7 GRAPHICS"

#### end of graphics DO_GRAPHICS
fi


##########################################
#   Module 8 Make the CORMS FLAGS
##########################################

echo " Make the cormsflags and ftp it to CORMS central "
echo \$time_nowcastend=$time_nowcastend
echo $time_nowcastend >> $MODELINFO/timetest.dat
# next line added to see if it removes the white space from front of time_nowcastend
time_nowcastend=`datemath $time_nowcastend + 0 0 0 0 0`
echo $time_nowcastend >> $MODELINFO/timetest.dat
MAKECORMSFLAGS.sh "$time_nowcastend"


##############################################################################
#   MODULE 9 Remove old files from archives
##############################################################################

time PURGE.sh
echo "That is execution time for PURGE.sh"

date
echo " That's All Folks!  "
```