



Meteorological Development Laboratory



SLOSH

Sea, Lake, and Overland Surges from Hurricanes User & Technical Software Documentation

**Silver Spring, Md.
October 2006**

**U.S. DEPARTMENT OF
COMMERCE**

**National Oceanic and
Atmospheric Administration**

**National Weather
Service**

Forward

SLOSH was originally developed in the mid 1960's by Chester P. Jelesnianski from the Institute for Oceanography, ESSA (now NOAA), with collaboration by the Weather Bureau's (now National Weather Service) D. Lee Harris and Albion Taylor. Since then, methods, software, and the operational use have evolved by the efforts of many people. The preparation of this document was a team effort led by Mark A. McInerney of the Meteorological Development Laboratory. Those making significant contributions are listed below:

Wilson Shaffer
NWS Meteorological Development Laboratory

Arthur Taylor
NWS Meteorological Development Laboratory

Karen A. Sandell
NWS Meteorological Development Laboratory

Brian Jarvinen
NWS Tropical Prediction Center

Stephen Baig
NWS Tropical Prediction Center

Gloria Lockett
NWS Tropical Prediction Center

Joan David
NWS Tropical Prediction Center

LT Jennifer Pralgo
NWS Tropical Prediction Center

Jerry Putland
NOAA Atlantic Oceanographic and Meteorological Laboratory

Heather Hauser
NWS Eastern Region Headquarters

Harry R. Glahn
Director - NWS Meteorological Development Laboratory

TABLE OF CONTENTS

1.0	GENERAL OVERVIEW	6
2.0	SLOSH Development Environment	9
2.1	System Requirements	10
2.2	Building the Development Environment	10
2.2.1	Installing Cygwin Environment	10
3.0	SLOSH Model Software Architecture	14
3.1	“Base” Model Context Diagram	14
3.1.1	High Level Context Diagram	14
3.1.2	First Decomposition	15
3.1.3	Second Decomposition - Base Model & DOS Wrapper	16
3.1.4	Third Decomposition - slosh2.c	17
3.1.5	Fourth Decomposition - tendian.c, pack.c	18
3.1.6	Fifth Decomposition - complex.c, savellx.c	19
3.1.7	Sixth Decomposition – myUtil.c, myassert.c	20
3.1.8	Seventh Decomposition – tio3.c	21
3.1.9	Eighth Decomposition – cstart.c	22
3.1.10	Ninth Decomposition – runslhg.f, interface.f, smooth.f	23
3.2	Model Subroutine Call Diagram	24
3.2.1	First Decomposition – DOS Wrapper (C)	24
3.2.2	Second Decomposition – Base Model (FORTRAN)	25
3.3	Base Model Source Tree	25
3.3.1	Source file “runslhg.f” Subroutines	26
3.3.2	Source file “smooth.f” Subroutines	68
3.3.3	Source file “interface.f” Subroutines	71
3.4	DOS Wrapper Source Tree	77
3.4.1	Source file “clock.c / cloch.h” Subroutines	78
3.4.2	Source file “complex.c / complex.h” Subroutines	90
3.4.3	Source file “cstart.c” Subroutines	96
3.4.4	Source file “myassert.c / myassert.h” Subroutines	99
3.4.5	Source file “myutil.c / myutil.h” Subroutines	100
3.4.6	Source file “pack.c / pack.h” Subroutines	105
3.4.7	Source file “savellx.c / savellx.h” Subroutines	111
3.4.8	Source file “slosh2.c / slosh2.h” Subroutines	117
3.4.9	Source file “tendian.c / tendian.h” Subroutines	121
3.4.10	Source file “tio3.c / tio3.h” Subroutines	129
4.0	Executing the SLOSH MODEL	137
4.1	Execution Commands for SLOSH	137
4.2	Creating a TRK Data File	137
5.0	SLOSH DISPLAY PROGRAM GRAPHICAL USER INTERFACE	139
5.1	SLOSH sloshdsp.ini Edit Graphical User Interface	145
5.2	Select Storm	146
5.2.1	Display Historical SLOSH Model Data	146
5.2.2	Display MEOW SLOSH Model Data	148
5.2.3	Display MOM SLOSH Model Data	149
5.3	Inquire All	151

5.4	Animate Historical Rex Data	154
5.5	Importing a Rex File	157
5.6	Generating PCX Based Animated GIF Images	157
5.7	Astronomical Tide Prediction	160
5.8	Storm Surge Time History	166
5.8.1	Storm Surge Time History Features and Functionality	166
Appendix A.	Acronyms	169

LIST OF FIGURES

Figure 1	– Storm Surge, Tide, and Wave Representation	6
Figure 2	– Shallow (Left) and Steep (Right) Continental Shelf	6
Figure 3	– SLOSH Display Program Map and Grid Example	7
Figure 4	– Individual SLOSH Grid	7
Figure 5	– SLOSH Model Grid Representing Land and Sea Elevation Averages	8
Figure 6	– SLOSH Basin Chart	9
Figure 7	– System Requirements Chart	9
Figure 8	– High Level SLOSH Context Diagram	14
Figure 9	– First Decomposition SLOSH Context Diagram	15
Figure 10	– Second Decomposition SLOSH Context Diagram	16
Figure 11	– Third Decomposition SLOSH Context Diagram	17
Figure 12	– Fourth Decomposition SLOSH Context Diagram	18
Figure 13	– Fifth Decomposition SLOSH Context Diagram	19
Figure 14	– Sixth Decomposition SLOSH Context Diagram	20
Figure 15	– Seventh Decomposition SLOSH Context Diagram	21
Figure 16	– Eighth Decomposition SLOSH Context Diagram	22
Figure 17	– Ninth Decomposition SLOSH Context Diagram	23
Figure 18	– First Decomposition SLOSH Subroutine Call Diagram	24
Figure 19	– Second Decomposition SLOSH Subroutine Call Diagram	25
Figure 20	– Base SLOSH Modules/Overlays	27
Figure 21	– 5 Point Grid Scheme	48
Figure 22	– Surge Offset Computations From Momentum	49
Figure 23	– Channel Flow Equation	51
Figure 24	– Computed 1-Dimension Flow Across Channel Of Interlocking Squares	52
Figure 25	– Interior/Exterior Calculated Flow For Individual Grid Cell	53
Figure 26	– Elevation on Interior/Exterior Calculated Flow for Individual Grid Cells	53
Figure 27	– Momentum Boundary Between Land and Deep Water	58
Figure 28	– Single Grid Cell Smoothing (Filter Function)	65
Figure 29	– Multiple Grid Cell Smoothing (Filter Function)	66
Figure 30	– Single Grid Cell Smoothing (Smoothing Function)	70
Figure 31	– Multiple Grid Cell Smoothing (Smoothing Function)	70
Figure 32	– TRK File Break Down	138
Figure 33	– SLOSH Display Program Initial Graphical User Interface	139
Figure 34	– Change Color GUI	140
Figure 35	– Change- Basin Graphical User Interface	144
Figure 36	– sloshdsp.ini Edit User Interface	145
Figure 37	– SLOSH Historical Storm Data Chart	146
Figure 38	– Select Storm “Historical” Interface	147
Figure 39	– Select Storm “Historical” Display	148
Figure 40	– Select Storm “MEOW” Interface	148
Figure 41	– Select Storm “MOM” Interface	150
Figure 42	– Example: Cat 5 MOM Display for Biscayne Bay Basin	151
Figure 43	– Inquire All User Interface	152
Figure 44	– Inquire All, Grid Specific, Text Output	153

Figure 45 – SLOSH Historical Rex Datafile On Install CD	154
Figure 46 – SLOSH Display Program’s Animation Setup Interface	155
Figure 47 – SLOSH I/J Coordinates	156
Figure 48 – .txt Tide Data	160
Figure 49 – Astronomical Tide Prediction or Tide Display Program	161
Figure 50 – Modify Graph GUI	162
Figure 51 – Tide Data User Interface, View Data	162
Figure 52 – Tide Stations in SLOSH	164
Figure 53 – Change Current Station / Edit Station Tide Display Interface	164
Figure 54 – Tide Display Program Highlighting Inquire Functionality	165
Figure 55 – Time History Interface	166
Figure 56 – ASCII Text File format of Time History Program Data	167
Figure 57 – Modify Time History Graph Parameters Interface	167
Figure 58 – Saffir-Simpson Scale	169
Figure 59 – Continental Shelf	170
Figure 60 – Tidal Datums	171

1.0 GENERAL OVERVIEW

The National Weather Service (NWS) conducts research and issues storm surge forecasts for hurricane and extratropical storms that threaten U.S. coastlines. This is done in support of public interest; local, state, and federal emergency manager evacuation procedures, and in support of the NWS mission to protect life and property. The NWS's solution to better understand, forecast, and protect U.S. coastlines from storm surge is based on over 40 years of ongoing research and development of the SLOSH model. This document serves as the user and technical manual for the SLOSH.

Storm surge is water pushed toward the shore by wind generated from synoptic scale tropical cyclones, although of most interest to NWS operations is hurricane and extratropical systems. This advancing surge combines with normal tides to create a hurricane storm tide capable of increasing the mean water level to more than 25 feet along the shore or 27.8 feet as in the case of hurricane Katrina in 2005. In addition, wind driven waves can be superimposed onto the storm tide increasing water levels to a point of causing severe flooding for coastal areas, particularly when this coincides with the normal high tides – figure 1.

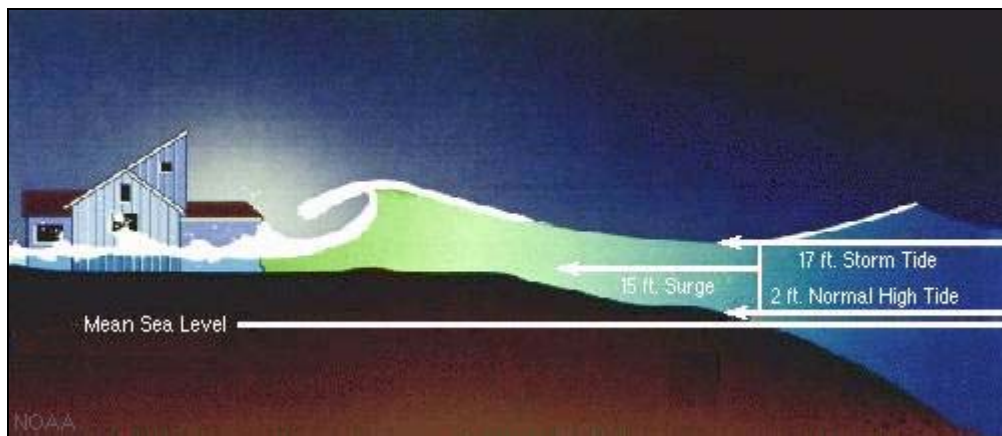


Figure 1- Storm Surge, Tide, and Wave Representation

The level of surge is also determined by the slope of the continental shelf as shown in figure 2. A shallow slope tends to produce a larger surge where a steeper continental shelf does not.



Figure 2 – Shallow (Left) and Steep (Right) Continental Shelf

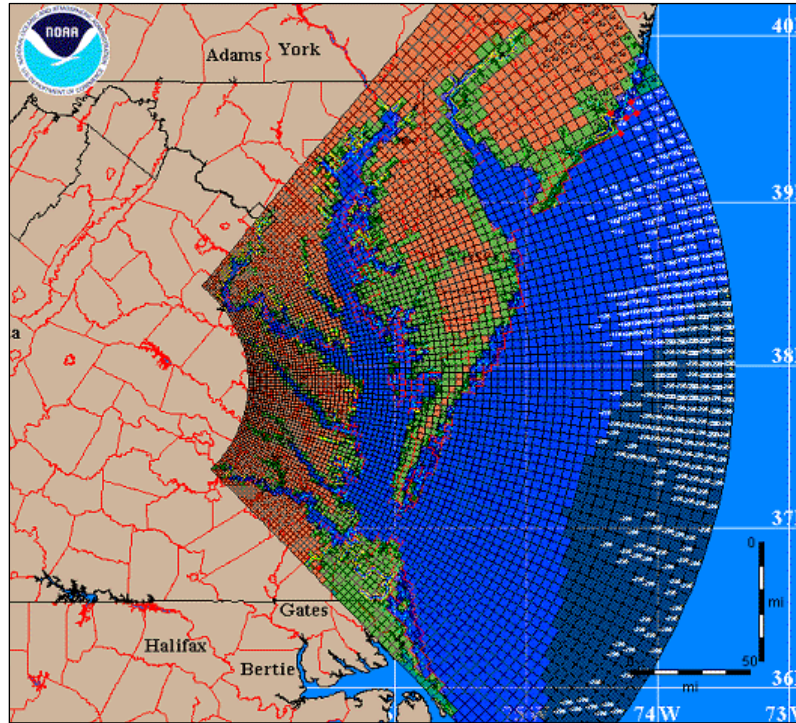


Figure 3 – SLOSH Display Program Map and Grid Example

SLOSH is a gridded based model where individual elements of the grid are used to calculate storm surge values along the coast and inland based on basin configuration – figure 3 and 6. The design contains barriers used in calculating water flow through each sub-grid element where water is allowed to completely flow: 1) through a barrier, 2) cuts in barriers, 3) one dimensional flow in rivers and streams, and 4) restricted due to trees and mangroves. The water depth is then calculated based on the elevation of the grid cell and the amount of water that is able to flow into that cell – see figure 4.

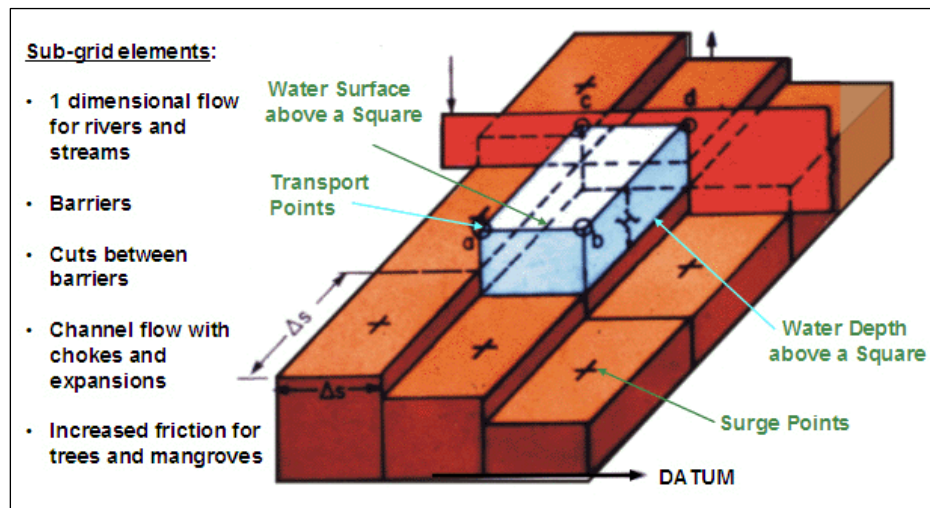


Figure 4 – Individual SLOSH Grid

The elevation for each grid cell is the average of the ground surface elevation found in a particular cell – see figure 5. This is represented by a number in the center of each cell and the water surface elevation is determined the same manner.

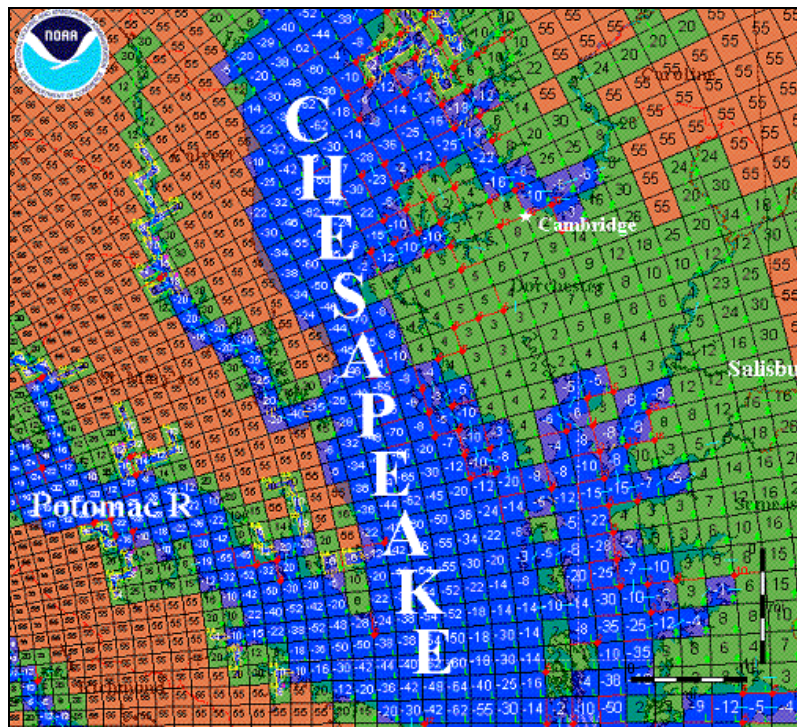


Figure 5 – SLOSH Model Grid Representing Land and Sea Elevation Averages

Operationally, SLOSH is used to forecast storm surge values which are incorporated into National Hurricane Center (NHC) forecast products. Moreover, the NHC also conducts simulation studies for evacuation planning used by emergency managers. Such studies are based on thousands of hypothetical hurricane tracks where the resulting water levels are categorized and stored by basin – for basin chart see figure 6. From these tracks, composite imagery is generated for each basin representing the Maximum Envelope of Water (MEOW). A MEOW is based on the highest surge values which include storm direction, forward speed, and simulated tide level at individual grid points for each storm category from the Saffir-Simpson Hurricane Scale. – Appendix A. In similar respect, Maximum of MEOWs, or MOMs, are composites generated from the maximum storm surge height for all hurricanes of a given category. MOMs also do not include forward speed, landfall direction, and land fall location.

For a more in-depth *scientific* understanding of SLOSH and storm surge readers are encouraged to reference:

- C.P.Jelesnianski, J.Chen, and W.A.Shaffer, 1992: SLOSH: Sea, Lake, and Overland Surges from Hurricanes, NOAA Technical Report NWS 48, April 1992, Silver Spring, Maryland.

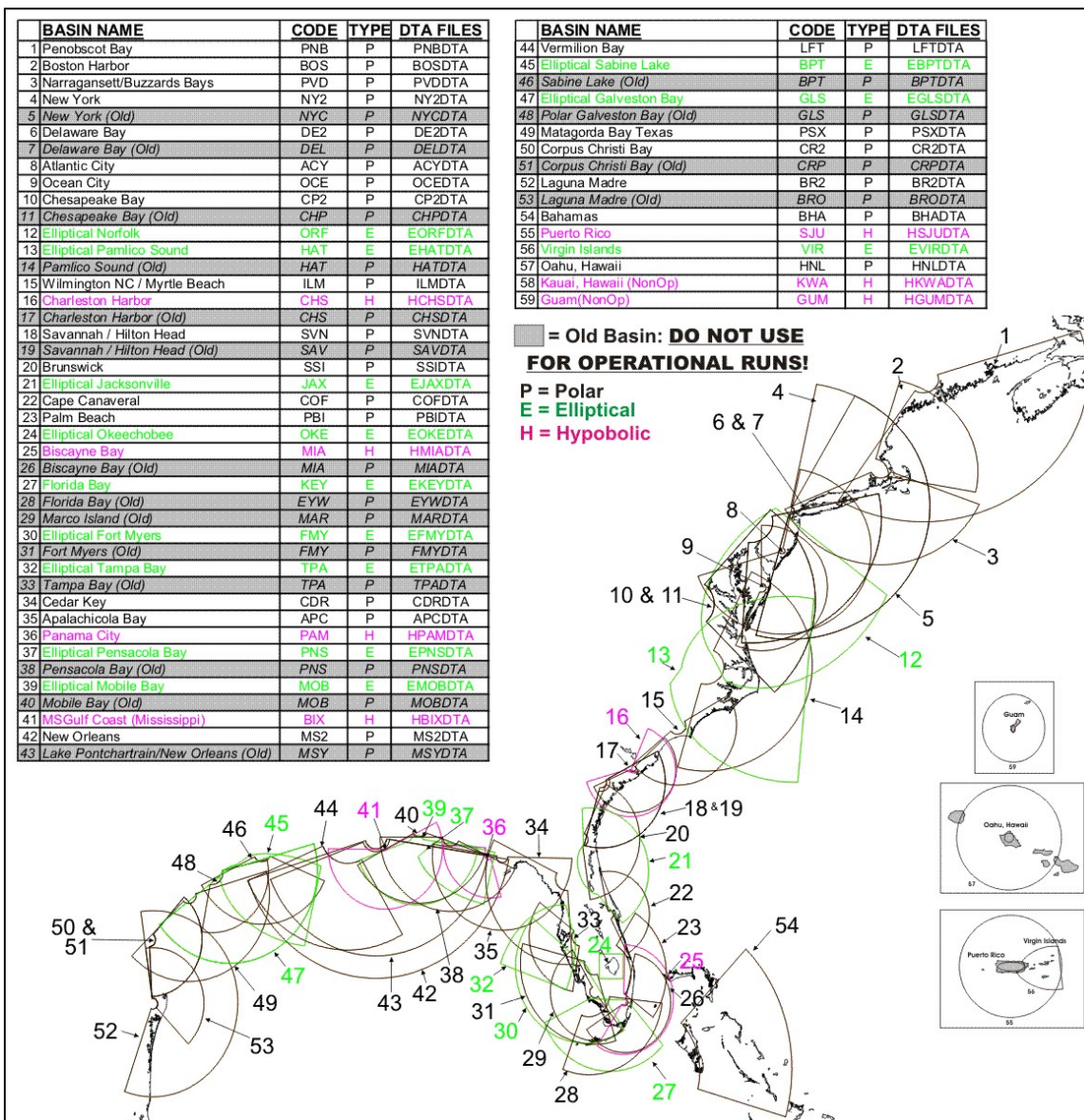


Figure 6 – SLOSH Basin Chart

2.0 SLOSH DEVELOPMENT ENVIRONMENT

This section describes in detail how to configure, execute, and utilize the SLOSH command line (DOS) based forecast model. The subsequent chapters provide the needed detail, however as a quick summary there are the major components required by the development environment.

Requirement	Solution	More Info
Operating System	Windows XP	www.Microsoft.com
Compilers (C/Fortran)	MinGW	www.MinGW.org
Development Environment	Cygwin	www.cygwin.com

Figure 7 – System Requirements Chart

2.1 System Requirements

Recommended system requirements are as follows, however no endorsement of specific software is expressed or implied:

- Microsoft Windows XP Professional or Home Edition
- Intel Pentium processor
- 256MB of RAM
- 4GB of available hard-disk space
- Microsoft Internet Explorer 5.5 (or higher), Netscape 7.1 (or 8.0), Firefox 1.0, or Mozilla 1.7

2.2 Building the Development Environment

The first step in building the SLOSH development environment is to create a SLOSH directory and unbundled the provided SLOSH software. The recommended location, name, and instructions are:

- 1) Open Windows Explorer interface
 - ▶ **Right click on *start* from the XP taskbar, then left click on *Explore***
- 2) Create SLOSH folder on root (EG, C) drive:
 - ▶ In the *Explorer* interface **left click on the root drive (C:) in the left navigation window**
 - ▶ In the right pain of *Windows Explorer* **right click on background of the right window pane** to display dropdown menu
 - ▶ In the displayed dropdown menu select: **New->Folder**
 - ▶ Name folder: **slosh**
- 3) Copy SLOSH_FOIAv1.0.zip bundle into created SLOSH directory
- 4) Unbundle SLOSH_FOIAv1.0.zip using your favorite unzip software.
 - Note: Most systems come with software to bundle or unbundle .zip files.*
 - However, if software is needed, a popular application is winzip and can be downloaded at <http://www.winzip.com>*

2.2.1 Installing Cygwin Environment

The latest instructions to install the Cygwin development environment can be found at <http://www.cygwin.com> or <http://cygwin.com/cygwin-ug-net/setup-net.html>. At the time of creating this document the Cygwin install instructions are as follows:

Internet Setup

To install the Cygwin net release, go to <http://cygwin.com/> and click on "[Install Cygwin Now!](#)". This will download a GUI installer called **setup.exe** which can be run to

download a complete cygwin installation via the internet. Follow the instructions on each screen to install Cygwin.

The **setup.exe** installer is designed to be easy for new users to understand while remaining flexible for the experienced. The volunteer development team is constantly working on **setup.exe**; before requesting a new feature, check the wishlist in the [CVS README](#) . It may already be present in the CVS version!

Since the default value for each option is the logical choice for most installations, you can get a working minimal Cygwin environment installed by simply clicking the `Next` button at each page. The only exception to this is choosing a Cygwin mirror, which you can choose by experimenting with those listed at <http://cygwin.com/mirrors.html> . For more details about each of page of the **setup.exe** installation, read on below. Please note that this guide assumes that you have a basic understanding of Unix (or a Unix-like OS). If you are new to Unix, you will also want to make use of [other resources](#) .

Download Source

Cygwin uses packages to manage installing various software. When the default `Install from Internet` option is chosen, **setup.exe** creates a local directory to store the packages before actually installing the contents. `Download from Internet` performs only the first part (storing the packages locally), while `Install from Local Directory` performs only the second (installing the contents of the packages).

The `Download from Internet` option is mainly for creating a base Cygwin package tree on one computer for installation on several machines with `Install from Local Directory`; copy the entire local package tree to another machine with the directory tree intact. For example, you might create a `C:\cache\` directory and place **setup.exe** in it. Run **setup.exe** to `Install from Internet` OR `Download from Internet`, then copy the whole `C:\cache\` to each machine and instead choose `Install from Local Directory`. Unfortunately **setup.exe** does not yet support unattended installs.

Though this provides some basic mirroring functionality, if you are managing a wide Cygwin installation, to keep up to date we recommend using a mirroring tool such as **wget**. A helpful user on the Cygwin mailing list created a simple demonstration script to accomplish this; search the list for **mkcygwget** for ideas.

Selecting an Install Directory

The `Root Directory` for Cygwin (default `C:\cygwin`) will become `/` within your Cygwin installation. You must have write access to the parent directory, and any ACLs on the parent directory will determine access to installed files.

The `Install For` options of `All Users` OR `Just Me` should always be left on the default `All Users`, unless you do not have write access to `HKEY_LOCAL_MACHINE` in the registry or the `All Users Start Menu`. This is true even if you are the only user planning to use

Cygwin on the machine. Selecting `Just Me` will cause problems for programs such as **crond** and **sshd**. If you do not have the necessary permissions, but still want to use these programs, consult the Cygwin mailing list archives about others' experiences.

The `Default Text File Type` should be left on `Unix` (that is, `\n`) unless you have a very good reason to switch it to `DOS` (that is, `\r\n`).

Local Package Directory

The `Local Package Directory` is the cache where **setup.exe** stores the packages before they are installed. The cache must not be the same folder as the Cygwin root. Within the cache, a separate directory is created for each Cygwin mirror, which allows **setup.exe** to use multiple mirrors and custom packages. After installing Cygwin, the cache is no longer necessary, but you may want to retain the packages as backups, for installing Cygwin to another system, or in case you need to reinstall a package.

Connection Method

The `Direct Connection` method of downloading will directly download the packages, while the `IE5` method will leverage your IE5 cache for performance. If your organization uses a proxy server or auto-configuration scripts, the `IE5` method also uses these settings. If you have a proxy server, you can manually type it into the `Use Proxy` section. Unfortunately, **setup.exe** does not currently support password authorization for proxy servers.

Choosing Mirrors

Since there is no way of knowing from where you will be downloading Cygwin, you need to choose at least one mirror site. Cygwin mirrors are geographically distributed around the world; check the list at <http://cygwin.com/mirrors.html> to find one near you. You can select multiple mirrors by holding down `CTRL` and clicking on each one. If you have the URL of an unlisted mirror (for example, if your organization has an internal Cygwin mirror) you can add it.

Choosing Packages

For each selected mirror site, **setup.exe** downloads a small text file called `setup.bz2` that contains a list of packages available from that site along with some basic information about each package which **setup.exe** parses and uses to create the chooser window. For details about the format of this file, see the [setup.exe homepage](#).

The chooser is the most complex part of **setup.exe**. Packages are grouped into categories, and one package may belong to multiple categories (assigned by the volunteer package maintainer). Each package can be found under any of those categories in the heirarchial chooser view. By default, **setup.exe** will install only the packages in the `Base` category and their dependencies, resulting in a minimal Cygwin installation. However, this will

not include many commonly used tools such as **gcc** (which you will find in the `Devel` category). Since **setup.exe** automatically selects dependencies, be careful not to unselect any required packages. In particular, everything in the `Base` category is required.

You can change **setup.exe**'s view style, which is helpful if you know the name of a package you want to install but not which category it is in. Click on the `View` button and it will rotate between `Category` (the default), `Full` (all packages), and `Partial` (only packages to be upgraded). If you are familiar with Unix, you will probably want to at least glance through the `Full` listing for your favorite tools.

Once you have an existing Cygwin installation, the **setup.exe** chooser is also used to manage your Cygwin installation. Information on installed packages is kept in the `/etc/setup/` directory of your Cygwin installation; if **setup.exe** cannot find this directory it will act just like you had no Cygwin installation. If **setup.exe** finds a newer version of an installed package available, it will automatically mark it to be upgraded. To `Uninstall`, `Reinstall`, or get the `Source` for an existing package, click on `Keep` to toggle it. Also, to avoid the need to reboot after upgrading, make sure to close all Cygwin windows and stop all Cygwin processes before **setup.exe** begins to install the upgraded package.

The final feature of the **setup.exe** chooser is for `Previous` and `Experimental` packages. By default the chooser shows only the current version of each package, though mirrors have at least one previous version and occasionally there is a testing or beta version of a package available. To see these package, click on the `Prev` or `Exp` radio button. Be warned, however, that the next time you run **setup.exe** it will try to replace old or experimental versions with the current stable version.

Download and Installation Progress

First, **setup.exe** will download all selected packages to the local directory chosen earlier. Before installing, **setup.exe** performs a checksum on each package. If the local directory is a slow medium (such as a network drive) this can take a long time. During the download and installation, **setup.exe** show progress bars for the current task and total remaining disk space.

Icons

You may choose to install shortcuts on the Desktop and/or Start Menu to start a `bash` shell. If you prefer to use a different shell or the native Windows version of `rxvt`, you can use these shortcuts as a guide to creating your own.

Post-Install Scripts

Last of all, **setup.exe** will run any post-install scripts to finish correctly setting up installed packages. Since each script is run separately, several windows may pop up. If you are interested in what is being done, see the Cygwin Package Contributor's Guide at

<http://cygwin.com/setup.html> When the last post-install script is completed, **setup.exe** will display a box announcing the completion. A few packages, such as the OpenSSH server, require some manual site-specific configuration. Relevant documentation can be found in the `/usr/doc/Cygwin/` or `/usr/share/doc/Cygwin/` directory.

3.0 SLOSH MODEL SOFTWARE ARCHITECTURE

3.1 Model Context Diagram

The context diagrams presented in this section were developed from existing slosh software, associated by function calls and returns.

3.1.1 High Level Context Diagram

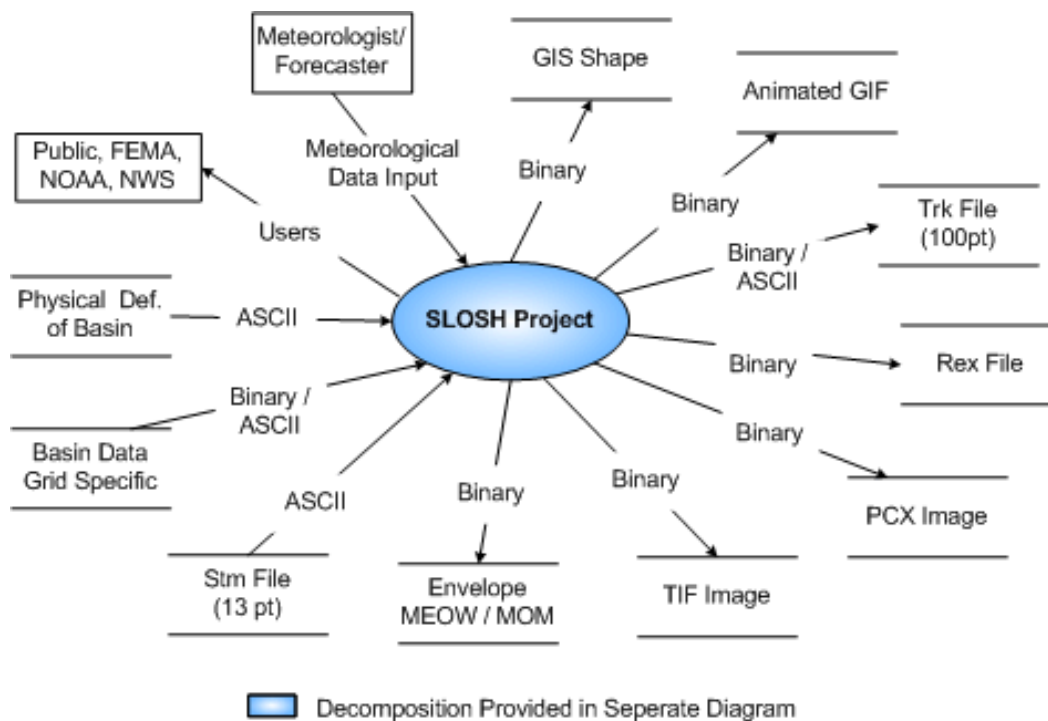


Figure 8 – High Level SLOSH Context Diagram

3.1.2

First Decomposition

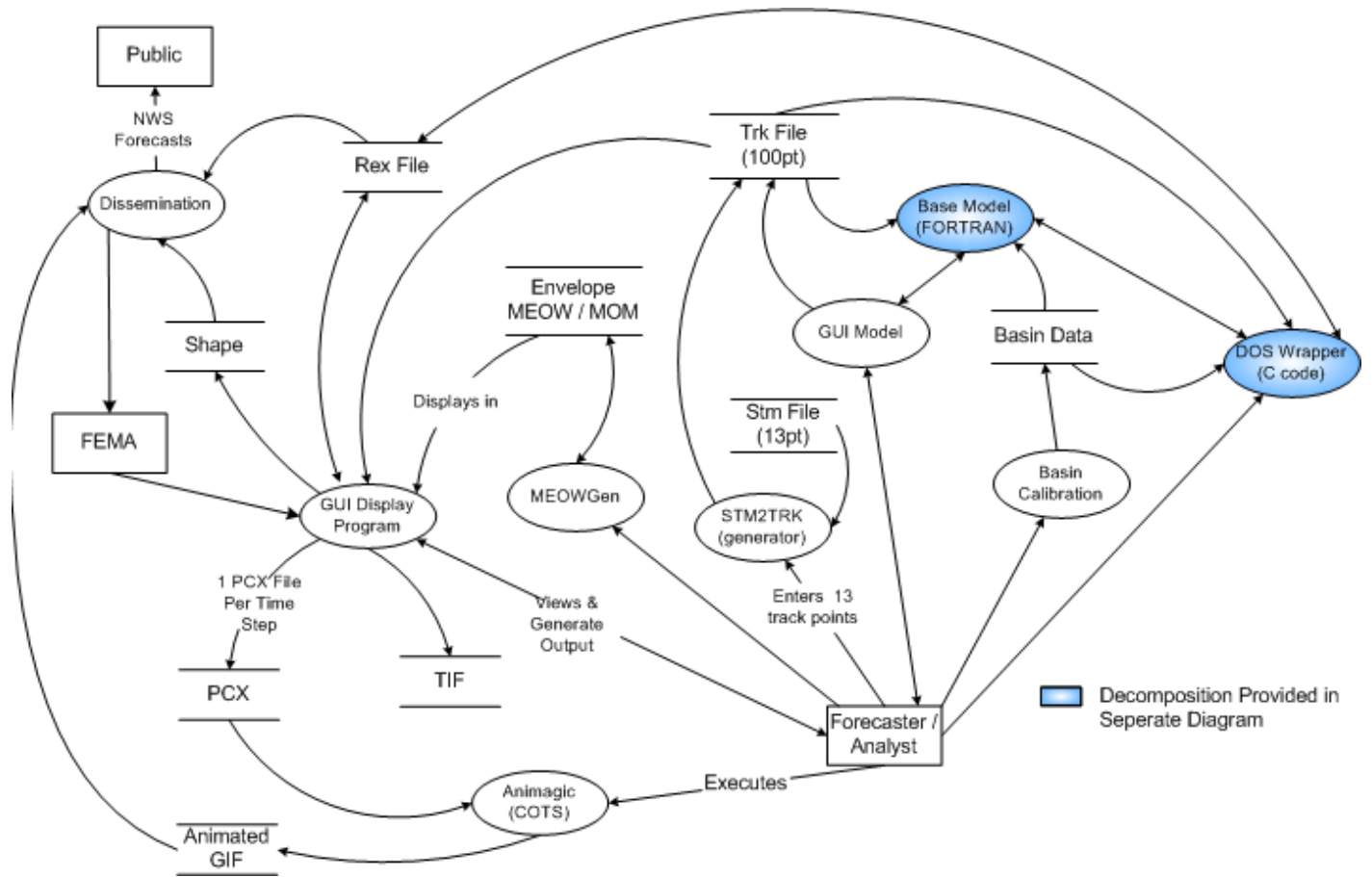


Figure 9 – First Decomposition SLOSH Context Diagram

3.1.3

Second Decomposition – Base Model & DOS Wrapper

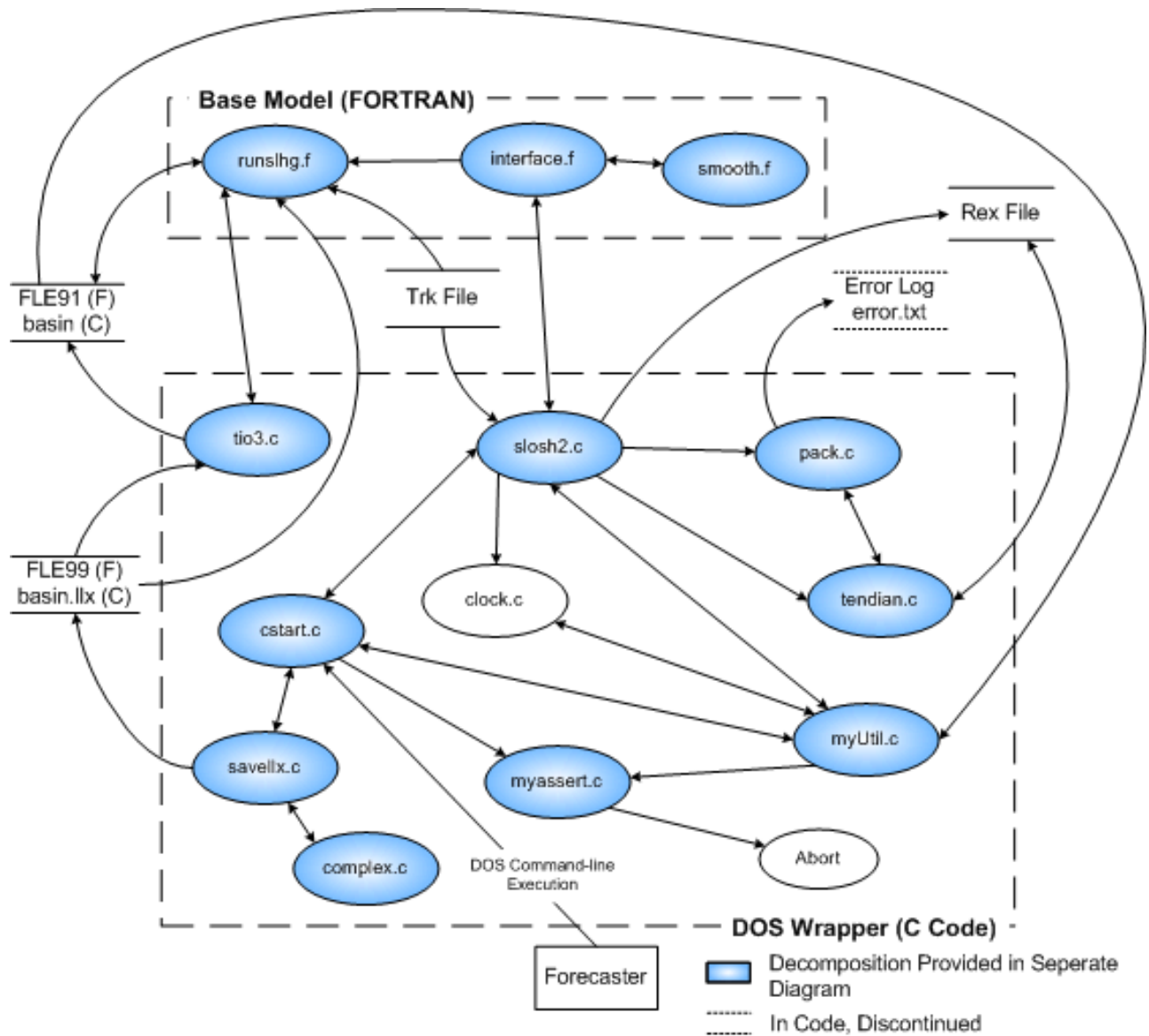


Figure 10 – Second Decomposition SLOSH Context Diagram

3.1.4

Third Decomposition – slosh.c

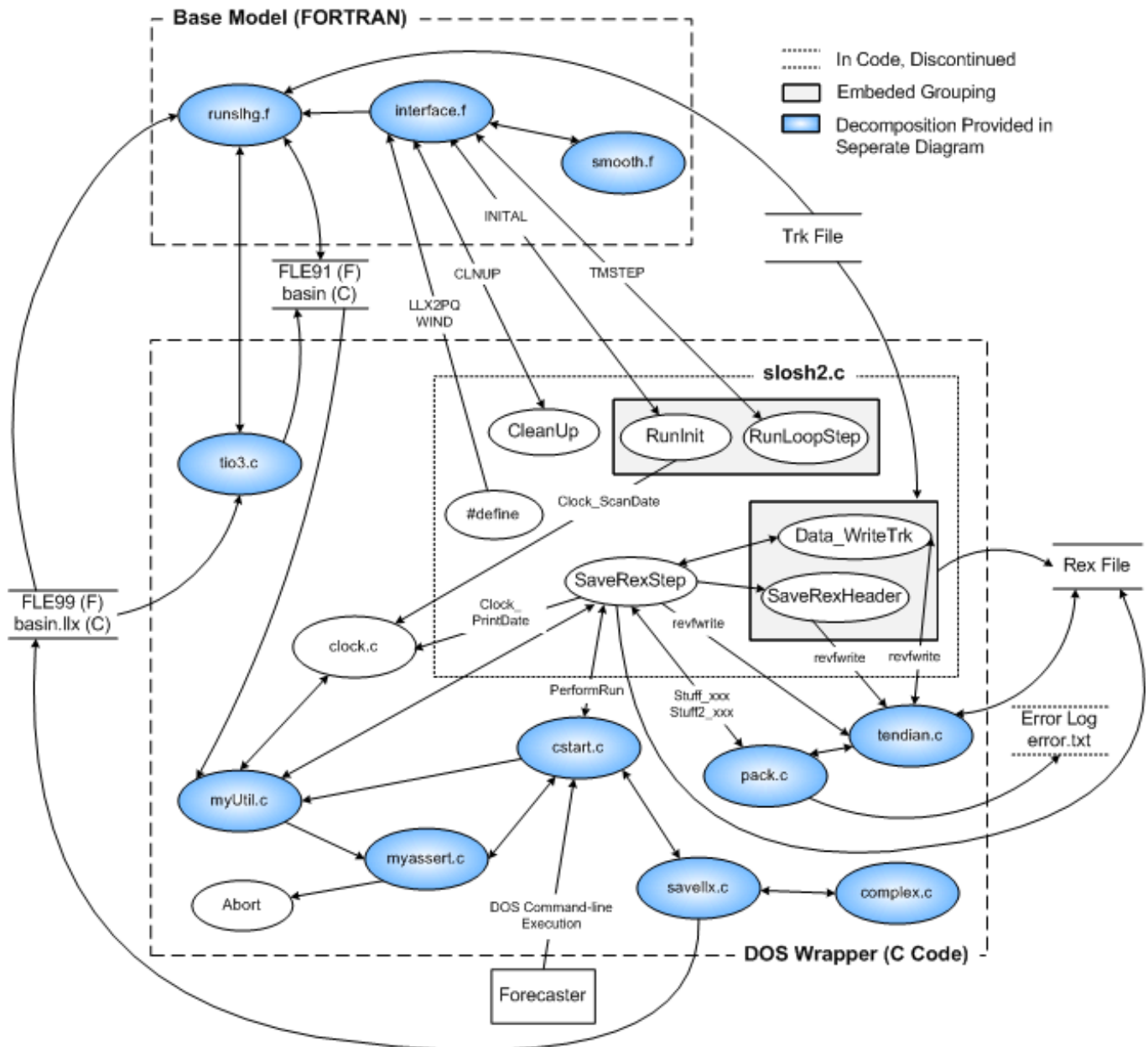


Figure 11 – Third Decomposition SLOSH Context Diagram

3.1.5

Fourth Decomposition – tendian.c, pack.c

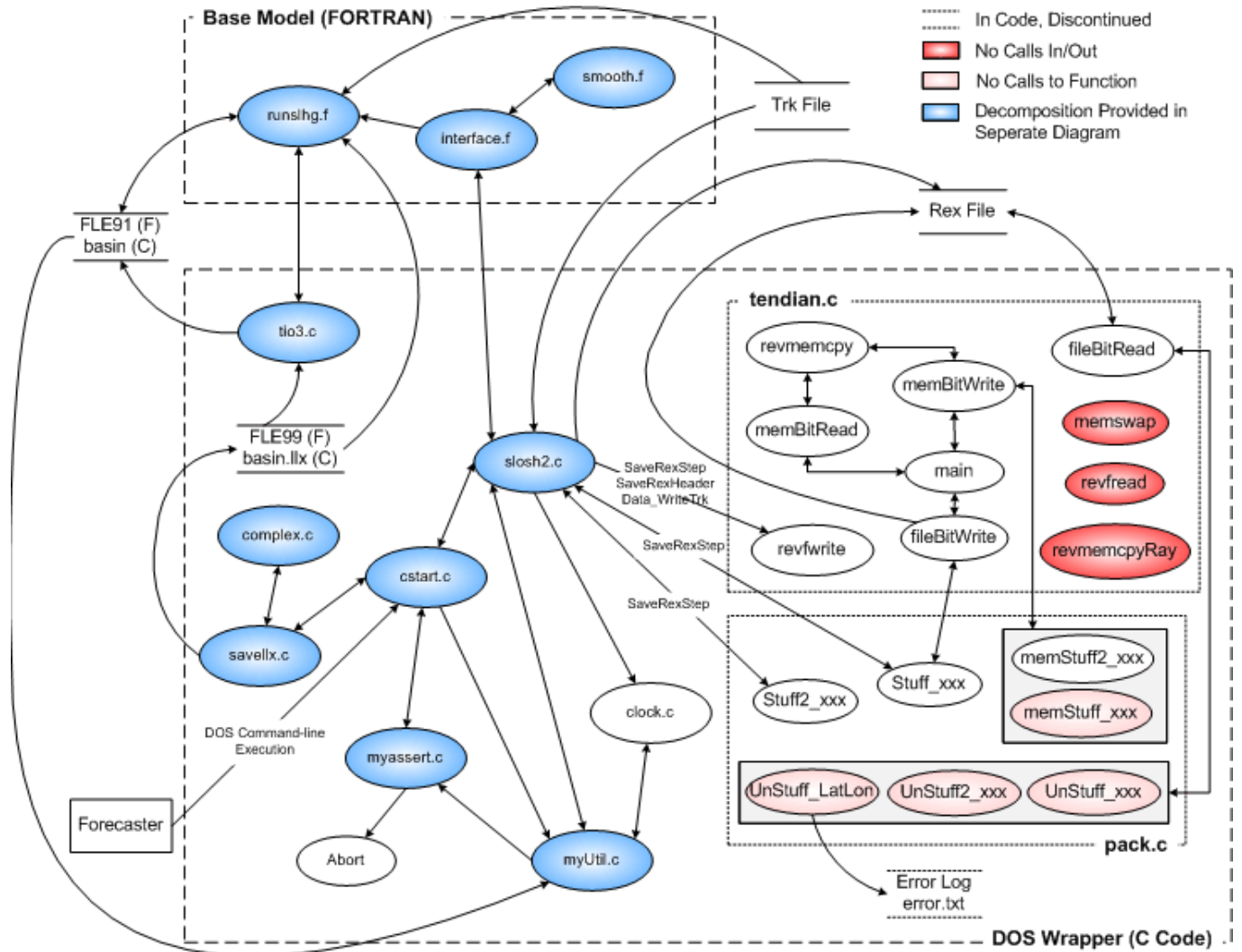


Figure 12 – Fourth Decomposition SLOSH Context Diagram

3.1.6

Fifth Decomposition – complex, savellx.c

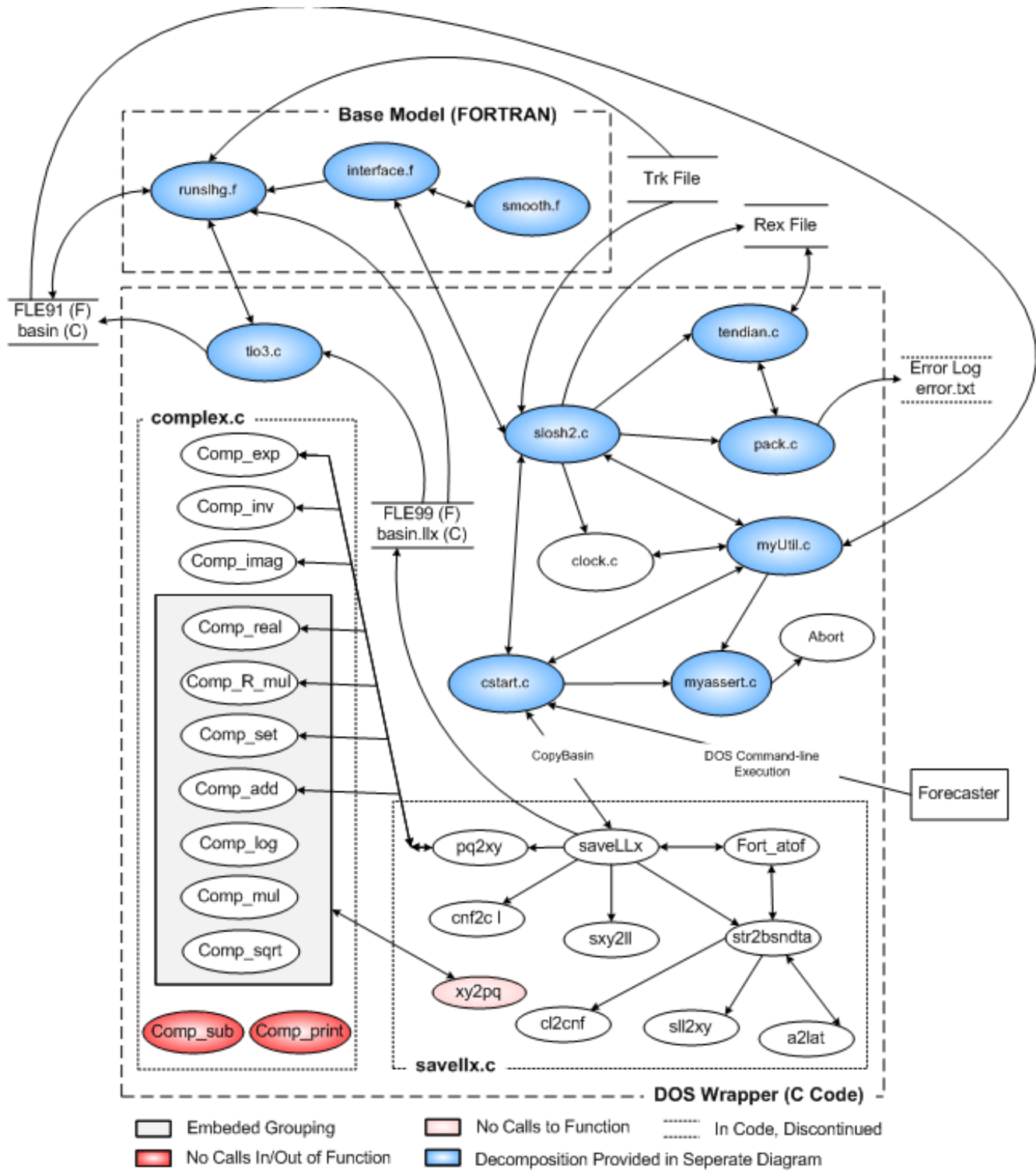


Figure 13 – Fifth Decomposition SLOSH Context Diagram

3.1.7

Sixth Decomposition – myUtil.c, myassert.c

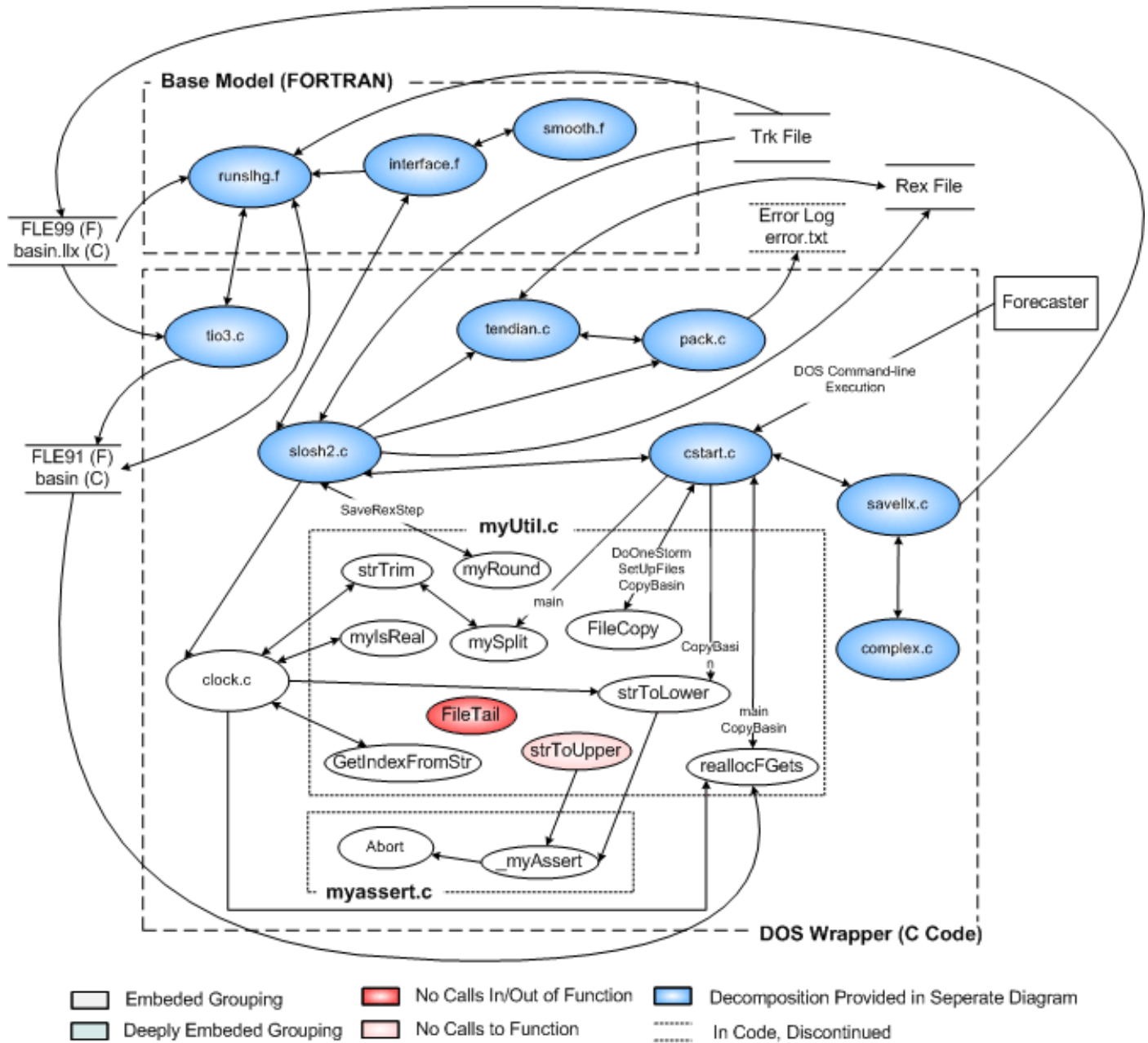


Figure 14 – Sixth Decomposition SLOSH Context Diagram

3.1.8 Seventh Decomposition – tio3.c

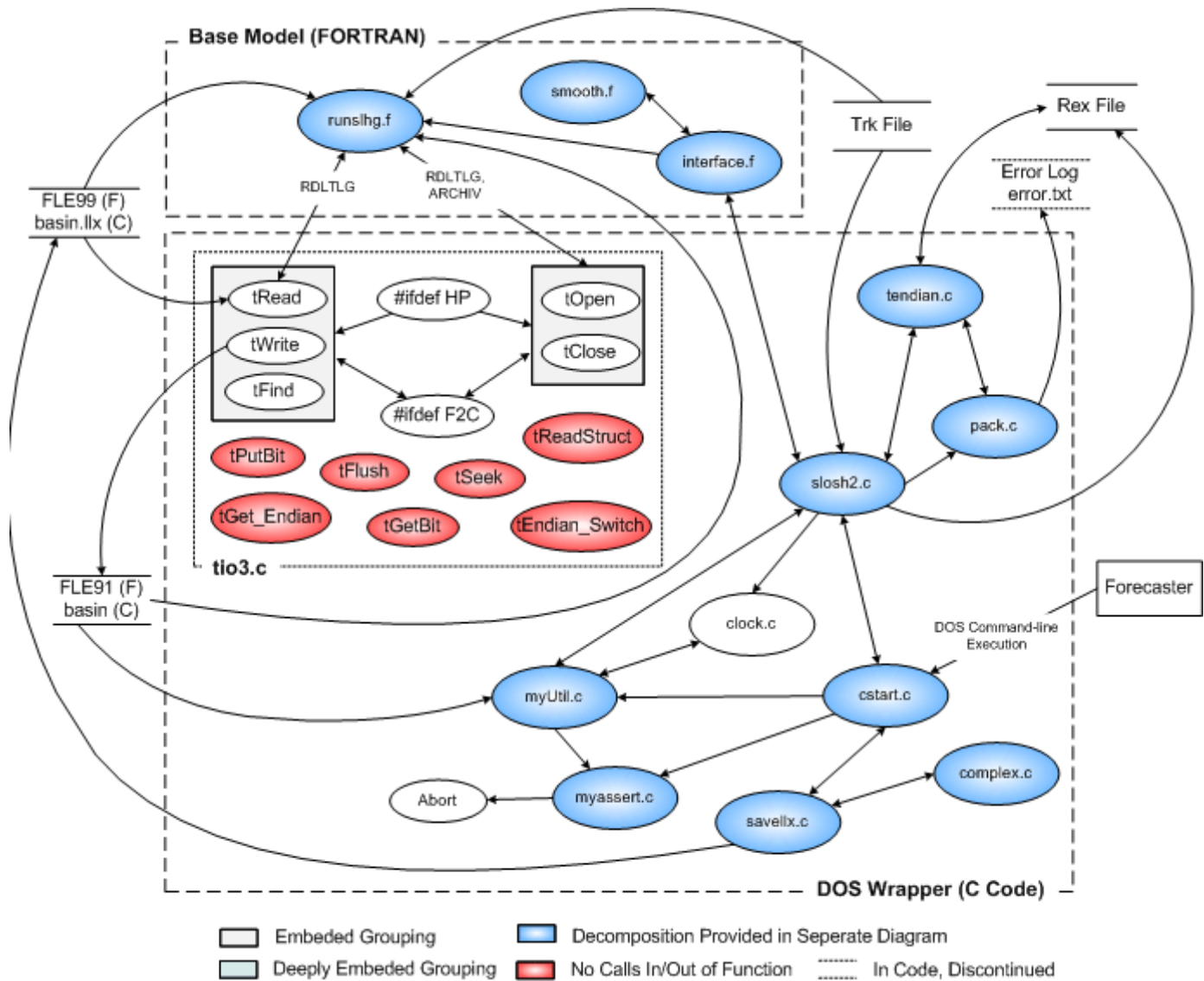


Figure 15 – Seventh Decomposition SLOSH Context Diagram

3.1.9

Eighth Decomposition – cstart.c

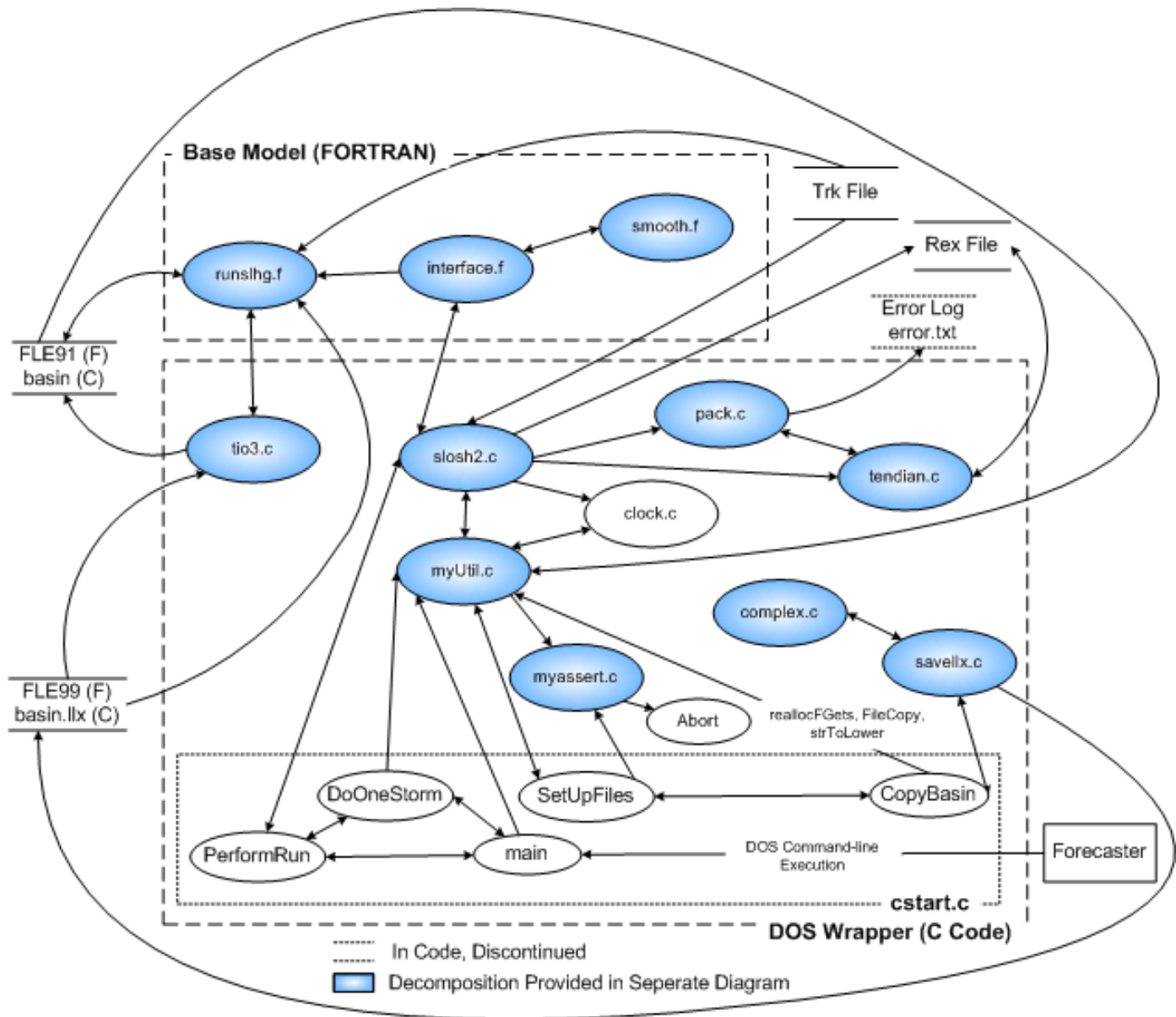


Figure 16 – Eighth Decomposition SLOSH Context Diagram

3.1.10

Ninth Decomposition – runslhg.f, interface.f, smooth.f

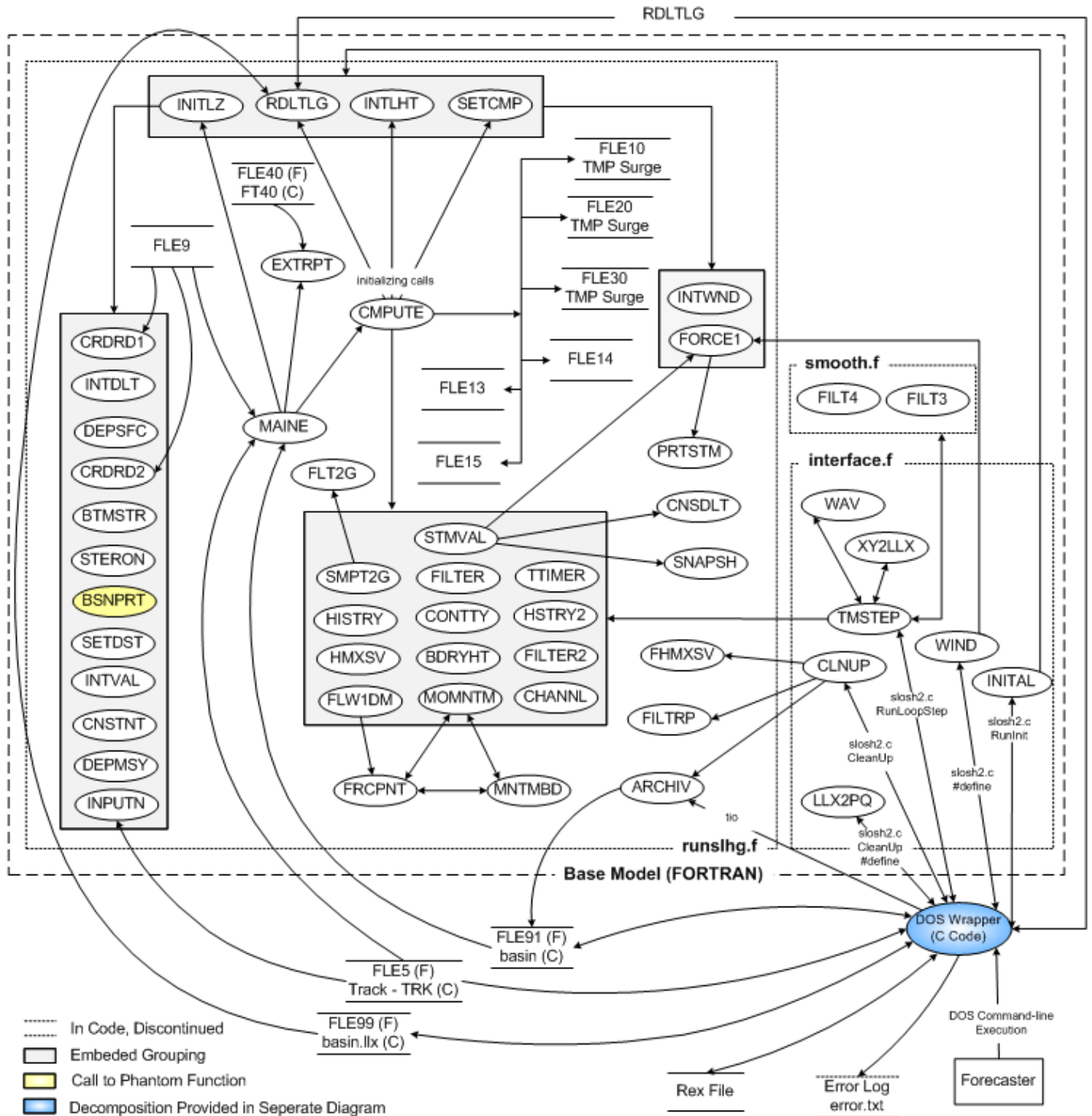


Figure 17 – Ninth Decomposition SLOSH Context Diagram

3.2 Subroutine Call Diagram

3.2.1 First Decomposition – DOS Wrapper (C)

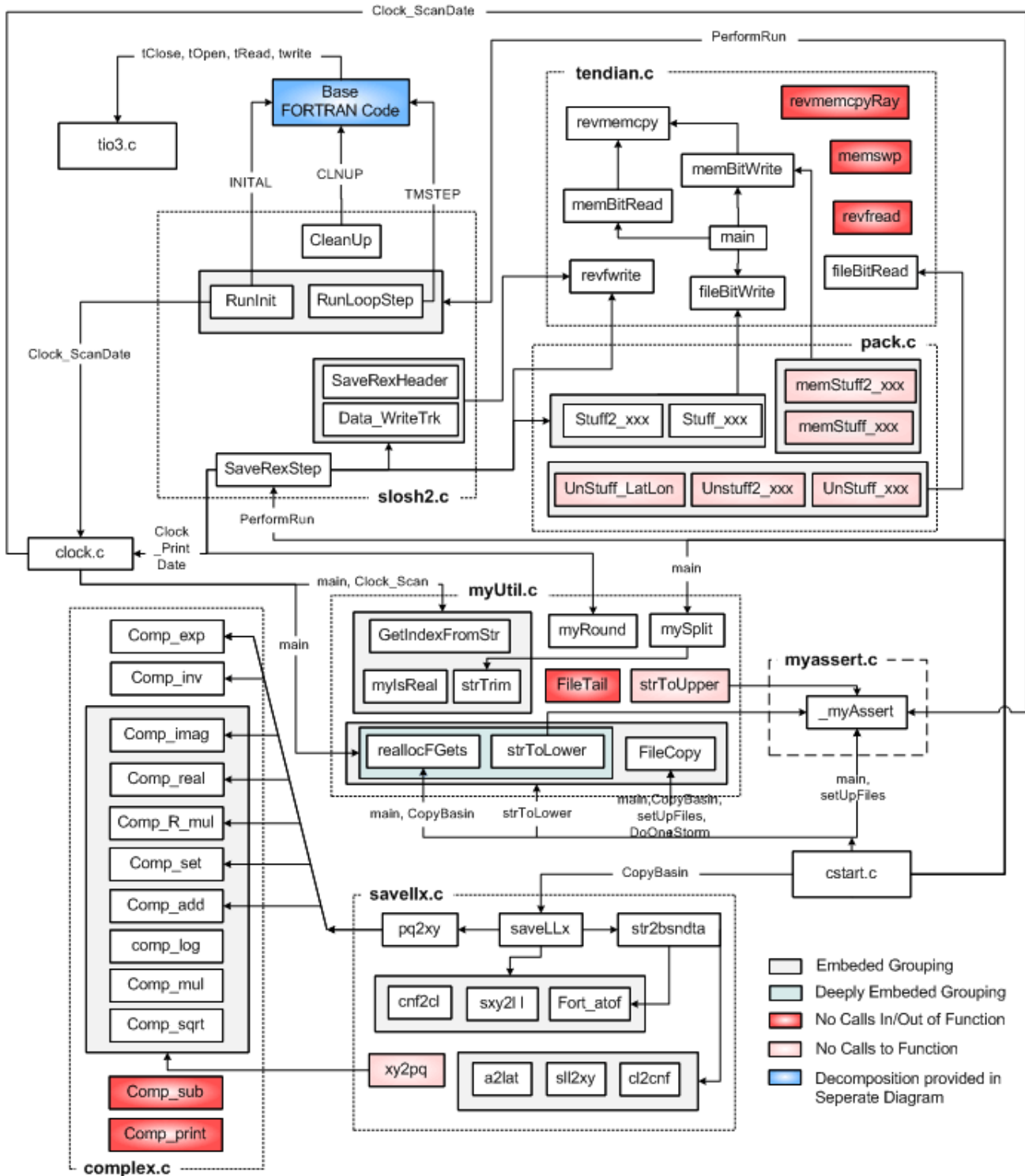


Figure 18 – First Decomposition SLOSH Subroutine Diagram

3.3.1

Source file “runslhg.f” Sub-function Breakdown

runslhg.f:

Author

Jelesnianski/Chen, July 1998 TDL/MDL

Source

FORTRAN 77

Purpose

Data Initialization / Global Variables

Comments

This member resides in the root overlay

Includes

parm.for

Variables Passed In

None

Local Variables

IP(4) JP(4) : Shift subscripts to 4 adjacent momentum points
IH(4) JH(4) : Shift subscripts to 4 adjacent surge points about a
momentum point
IIH(4) JJH(4) : Shift subscripts to 4 adjacent surge points about a surge
point.
IOPERL : 2 as default in operational mode
MAPIN(HR) : Specifies the print hour, before and after the nearest
approach of storm, for snapshot output
G : Gravity
C7 : Slip coefficient
C17 : Air density
C19 : Friction coefficient
C25 : Eddy viscosity
MONTH(12) : Accumulated days at monthly increments (-1, 30, 58, 89,
119 ...)

Return

None

runslhg.f: Subroutines

MAINE:

Author

Jelesnianski/Chen, July 1992 TDL/MDL

Purpose

This is the main program, also called root overlay which is calls 4 overlays as shown, member name is *main*.

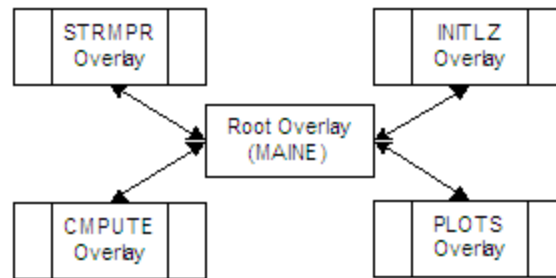


Figure 20 – Base SLOSH Modules/Overlays

Comments

- 1) This code works in conjunction with VSGL0T21.FOR to generate load module for evacuation study. Let (MAIN, PLOTS, CMPUTE) supersede those in VSGL0T21.FOR.
- 2) Major Changes include 1) Surge histories at 10N grid points, 2) Input format for hourly press, and 3) radius archive max of surges & max winds.
- 3) If no archive points are given (KHSPT=0), NOARCH takes the default value, X. In a non-archived run default 10 points and time histories are in 10-15 minute intervals.
- 4) If archive envelope is desired use blank data for *Archive Points*. This will then overwrite the default value of X NOARCH to NULL, telling the software *ARCHIEV* is active.

Data Set Use

None

Includes

parm.for

Variables Passed In

None

Local Variables

NA

Subroutines Calling Function (File:Subroutine):

None

Subroutine Calls (File:Subroutines)

runslhg.f: NITLZ, EXTRP, COMPUTE

Return

None

EXTRPT:

Author

Unknown

Purpose

Unknown

Comments

None

Data Set Use

None

Variables Passed In

FILE40 : Unknown

Local Variables

TBD

Subroutines Calling Function (File:Subroutine):

runslhg.f: MAINE

Subroutine Calls (File:Subroutines)

None

Return

None

INTDLT:

Author

Jelesnianski, August 1980 TDL/MDL

Purpose

Subroutine ascertains time step *DELT* for computations, initializes storm at subscript *IBGNT* and sets *INEND* for ending computations after nearest storm approach.

Comments

None

Data Set Use

None

Variables Passed In

None

Local Variables

DELT : Time interval for explicit finite differencing
 DLTB : The smallest reduction of *DELT* in model run
 RLNGTH : Distance of storm from nearest approach
 JHR : Subscript for nearest approach to basin center
 P(100) : Hourly pressure drops
 X(100) Y(100): Hourly components for storm position
 R(100) : Hourly storm sizes
 XMOUTH : X-distance from pole to basin center
 YMOUTH : Y-distance (usually equal to 0)
 IBGNT : Tabulated subscript for storm initialization
 ITEND : Tabulated subscript for storm termination
 IBGS : Number of hours for computations before nearest storm approach
 ITMS : Number of remaining hours, after nearest approach
 DLTINS(3) : Time steps for 3 categories of storms from data

Subroutines Calling Function (File:Subroutine):

runslhg.f: INITLZ

Subroutine Calls (File:Subroutines)

None

Return

None

INTVAL:

Author

Jelesnianski, August 1980 TDL/MDL

Purpose

This subroutine sets initial storm parameters; some fixed constants for initialization and output times

Comments

1) Use only time steps divisible by 600 s(10 min), or 900 s(15min)

Data Set Use

None

Variables Passed In

None

Local Variables

DELT : Time interval for explicit finite differencing
NDLTHR : Number of time steps in 1 hour
NDLTH2 : Number of time steps in ½ hour
IPRTSV : Time print of historical surges (10 points)
IBGNT : Subscript for storm initialization
SP(100) : Hourly forward storm speeds
DIR(100) : Hourly Storm azimuths
R(100) : Hourly storm sizes
IBGS : Number of hours for computations before nearest approach
ITMS : Number of hours for computations after nearest approach
MHALT : Number of time steps for total computations
ITEND : Tabulated subscript for storm termination
IC19 : Growth time to mature storm
MAPIN(10) : Printout time for envelope snapshot
IOPERL : Sets operational mode if value is “2”
ISTMAP : At this time, print a snapshot of surges
NUMBER : Closet hour to nearest approach

Subroutines Calling Function (File:Subroutine):

runslhg.f: INITLZ

Subroutine Calls (File:Subroutines)

None

Return

None

CNSTNT:**Author**

Jelesnianski, August 1980 TDL/MDL

Purpose

This Subroutine fixes some program constants

Comments

None

Data Set Use

None

Variables Passed In

None

Local Variables

X1(50) : array of constants

DELTA : Time increment for finite differencing, set in subroutine *INTDLT*
 COR : Coriolis (1/sec), set in subroutine *INTVAL*
 IMXB JMXB : Number of grid points along a ray/circle, in CRDRD1
 IMXB1 : IMXB less 1,
 JMXB1 : JMXB less 1
 IMXB2 : IMXB less 2,
 JMXB2 : JMXB less 2
 DEGREE : Angle between two rays, red in *CRDRRD1*
 DELA : Value of *DEGREE* in radians
 RAD : Radian measure
 X1B(10) : Array of constants
 G : Gravity = 32.2 FT/SEC-2
 AMAN->AGB13: Fixed constant for subroutine *CHANNL*

Subroutines Calling Function (File:Subroutine):

runslhg.h: INITLZ

Subroutine Calls (File:Subroutines)

None

Return

None

BTMSTR:

Author

Jelesnianski, August 1980 TDL/MDL

Purpose

Computes bottom stress coefficients at 1 foot intervals, from 1 to 300 feet.
 Equations are defined in: Jelesnianski, "Numerical Computations of Storm
 Surges with Bottom Stress", monthly weather review, VOL 95, NO. 11
 Nov 1967, pp 740-756

Comments

This member *BTMSTR* resides in overlay. Program rewritten (1980)
 replacing SINH by SIN and COSH by COS.

Data Set Use

None

Variables Passed In

ZLATO : Latitude in degrees

Local Variables

COR : Coriolis parameter

E : Ekman parameter, $DEPTH * \sqrt{COR/2 * C25}$

C25 : Eddy viscosity coefficient, .25 FT**2/SEC
 C7 : Slip coefficient, .006 FT/SEC
 AR(300) : Bottom friction coefficients for coriolis
 AI(300) : Bottom friction coefficients for coriolis terms
 BR(300) : Bottom friction coefficients for surface gradient terms
 BI(300) : Bottom friction coefficients for surface gradient terms
 CR(300) : Bottom friction coefficients for surface stress terms
 CI(300) : Bottom friction coefficients for surface stress terms

Subroutines Calling Function (File:Subroutine):

runslhg.f: INITLZ

Subroutine Calls (File:Subroutines)

None

Return

None

SETCMP:

Author

Jelesnianski, August 1980 TDL/MDL

Purpose

Sets the first hourly meteorological and track data for computations and also sets changes in data for the first incremental time step.

Comments

This subroutine resides in overlay *CMPUTE*.

Data Set Use

None

Variables Passed In

None

Local Variables

INTIME : Number of time steps; initialized to zero.
 NDLTR : Number of time steps in one hour
 HHRAD : Accumulative time steps on the hour
 IBGNT : Tabulated subscript for storm initialization
 PT(100) : Hourly pressure HROPS
 PDROP : Dummy value for 1st hour pressure drop
 YDELP : Pressure drop at present time step
 ZDELP : Pressure drop at present time step
 PN : Pressure drop at present time
 PNN : Pressure drop at previous time
 R(100) : Hourly storm sizes

YC24 : Storm size at present time step
 C24 : Storm size at previous time step
 C21 : Dummy value for speed of storm 1st hour C21 initially set TVAL
 X12(7) : Speed of storm in ft/sec, 1st hour
 X12(20) : Direction of storm motion for 1st hour, set in *INTVAL*
 X12(21) : Direction of storm motion for 1st hour, set in *INTVAL*
 X12(9) : Components of forward speed, 1st hour
 X12(10) : Components of forward speed, 1st hour
 DELTA : Unit time step
 X12(11) : Components of storm motion unit time step, 1st hour
 X12(12) : Components of storm motion unit time step, 1st hour
 C22 : Direction of storm motion, 1st hour, set in *INTVAL*
 X12(17) : Friction coefficient (3x10-6) times square of storm size
 X12(23) : Components of wind distortion (lake winds), 1st hour
 X12(24) : Components of wind distortion (lake winds), 1st hour
 PTENCY : Change in pressure drop for 1st hour
 RTENCY : Change in storm size for 1st hour
 ITMADV : Counter for hourly time, 1st hour
 AX AY : Components of total storm traverse with time
 W1218 : Twice the maximum winds (lake or ocean)
 C(800) S(800) : COS/SIN of inflow angles, mile intervals from storm center (sea winds), computed in FORCE1
 CW(800) : COS/SIN inflow angle (lake winds), from FORCE1
 SW(800) : COS/SIN inflow angle (lake winds), from FORCE1

Subroutines Calling Function (File:Subroutine):

runslhg.f: CMPUTE
 interfae.f: INITAL

Subroutine Calls (File:Subroutines)

runslhg.f: INTWND, FORCE1

Return

None

STMVAL:

Author:

Jelesnianski, August 1980 TDL/MDL

Purpose:

Tests for incremental hours and sets hourly track and meteorology values.
 Also tests for 1-MB change in pressure drop, or 1 mile change in storm size for each advance in time. Preparatory values are set for later advancing in time.

Comments

Subroutine resides in overlay *CMPUTE*.

Data Set Use

None

Variables Passed In

None

Local Variables

ITIME	: Cumulate time counter
N1	: Time steps at hourly intervals, set in SETCMP
NDLTHR	: Number of time steps in one hour
ITMADV	: Counter for hourly time set in SETCMP
C22	: Dummy value for hourly direction of storm motion
DIR(100)	: Hourly direction of storm movement
X12(20)	: Directions of storm motion, hourly value
X12(21)	: Directions of storm motion, hourly value
SP(100)	: Hourly forward speed of storm
C21	: Dummy value for storm's hourly forward speed
X12(7)	: Speed of storm in ft/sec, hourly values
X12(11)	: Components of forward storm speed
X12(10)	: Components of forward storm speed
DELTA	: Unit time step
X12(11)	: Storm movement in X/Y-DIR for unit time step
X12(12)	: Storm movement in X/Y-DIR for unit time step
X12(17)	: Friction coefficient (3×10^{-6}) Times square of Storm Size
PT(100)	: Hourly pressure HROPS
R(100)	: Hourly storm sizes
PTENCY	: Change in pressure drop for one hour
RTENCY	: Change in storm size for one hour
YC24	: Storm size for present time
YDELP	: Pressure drop for present time
AX AY	: Components of total storm traverse with time
ZDELP	: Pressure drop at previous MB step
PN	: Integer value of pressure drop at present time
PNN	: Integer value of pressure drop at previous time
WMAX(2)	: Max wind speed over inland water bodies or ocean
IGBNT	: Subscript for storm initialization
JHR	: Subscript for nearest storm approach
ZC24	: Storm size at previous mile step
VMAX	: Dummy value for maximum winds
W1218(2)	: Twice the maximum wind (lake or ocean)
C(800) S(800)	: Cosine-sine of inflow angle, mile intervals from storm center (sea winds)
CW(800)	: Cosine-sine of inflow angle (lake winds)
SW(800)	: Cosine-sine of inflow angle (lake winds)

ISTMAP : Printout time for snapshots, set in *BLCKDT*
 MAPIN(10) : Printout time for snapshots, set in *BLCKDT*
 Number : Subscript for printout of above set in *INTVAL*
 IPRHR : 1, when on the hour; 2, when half hour in between; 3, otherwise.
 IPRTSV : Time to save historical surges, 10 points, set *INTVAL*.
 IPRTAD : Update time of *IPRTSV*, set in *INTVAL*
 IOPERL : 2 if in operational mode

Subroutines Calling Function (File:Subroutine):

runslhg.f: CMPUTE
 interface.f: TMSTEP

Subroutine Calls (File:Subroutines)

runslhg.f: SNAPSH, CNSDLT, FORCE1

Return

None

CNSDLT:

Author

J Chen, August 1982 TDL/MDL

Purpose

Fixes coefficients depending on time step for variations in the course of computation

Comments

None

Data Set Use

None

Variables Passed In

None

Local Variables

X1(50) : Array of constants
 DELT : Time increment for finite differencing, set in INTDLT
 COR : Coriolis (1/sec), Set in INTVAL
 Degree : Angle between two rays, read in CRDRD1
 DELA : Value of DEGREE in radians
 RAD : Radian measure
 X1B(10) : Stored array of constants
 G : Gravity = 32.2 ft/sec-2

Subroutines Calling Function (File:Subroutine):

runslhg.f: STMVAL

Subroutine Calls (File:Subroutines)

None

Return

None

FORCE1:

Author

Jelesnianski, September 1980 TDL/MDL

Purpose

Computes at mile interfals from the storm center, for 800 miles

1. SINE of the inflow angle
2. COSINE of the inflow angle
3. Surface pressure
4. Static height

Comments

Subroutine resides in overlay *CMPUTE*.

Data Set Use

None

Variables Passed In

PDROP : Input pressure drop
RMAX : Dummy value for storm size
VMAX : Dummy value for maximum winds
ALTO : Latitude of basin mouth point
NCATG : TBD
IPRT : 1 to printout storm profile, otherwise = 0

Local Variables

XKS XKN : Wind friction values
CFRIC : Friction faction, = 1. In ocean, =4*SQRT(22/RMAX)
JSET : Distance beyond which approx by constant inflow angles
COR : Coriolis (1/sec)
C(800) S(800): COS/SIN of inflow angle, mile intervals
DELP(800) : Static heights, mile intervals
P(800) : Pressure gradient, mile intervals
RHOA : Air density (MB HR**2 MILE**-2)
RHOW : Water density (MB HR**2 FT**-1 MILE**-1)
RHOWG : Water density*gravity (MB ft**-1)
PADD : Pressure from a radius to infinity
PINFT : Pressure drop from iteration

Subroutines Calling Function (File:Subroutine):

runslhg.f: SETCMP, STMVAL

interface.f: WIND

Subroutine Calls (File:Subroutines)

runslhg.f: PRTSTM

Return

None

INTWND:**Author**

Jelesnianski, August 1980 TDL/MDL

Purpose

Sets up an initial guess for the max wind of a stationary storm using initial input storm parameters. The initial guess is an entry value to subroutine FORCE1 for iteration to exact maximum winds

Comments

- 1) This subroutine is in overlay STRMPR
- 2) For a given pressure drop P(100), the max wind varies almost linearly with storm size R(100)

The slope and intercept of the pressure line varies weakly with latitude or coriolis. The given WM, WB are for latitude 30 degrees; The entry pressure at time IBGNT is doctored to correct for latitude other than 30 degrees.

Data Set Use

None

Variables Passed In

PDROP : Pressure drop for 1st hour
RMAX : Radius of maximum wind
ALTO : Latitude of basin center
VMAX : Maximum wind

Local Variables

KPRES : Doctored pressure drop for lat other than 30 degrees
WM(150) : Array for slope of linear slope formula
WB(150) : Array for intercept of linear slope formula

Subroutines Calling Function (File:Subroutine):

runslhg.f: SETCMP

interface.f: WIND

Subroutine Calls (File:Subroutines)

None

Return

None

PRTSTM:

Author

Unknown

Purpose

Prints out, at the time of nearest approach, the two storm systems for lake and ocean. The storm values are incremented at regular intervals from the storm center.

Comments

Subroutine resides in overlay *CMPUTE*

Data Set Use

None

Variables Passed In

PADD : Pressure drop from a radius to infinity
PINFT : Pressure drop from iteration
VMAX : Dummy value for maximum winds
RMAX : Radius of maximum wind
ALTO : Latitude of basin center
NCATG : 1 for lake winds, =2 for ocean

Local Variables

P(800) : pressure gradient
S(800) : SINE of inflow angle
DELP(800) : Static heights from storm center
RHOWG : Water density*gravity (MB SEC**2 FT**-1 MILE**-1)
PHI : Inflow angle
IBEG(3) : Begin loop
IEND(3) : End loop
IN(3) : Increment of do loop

Subroutines Calling Function (File:Subroutine):

runslhg.f: FORCE1

Subroutine Calls (File:Subroutines)

None

Return

None

CRDRD1:

Author

Jye Chen, May 1990 TDL/MDL

Purpose

Reads in basin projection data, as input data to the SLOSH program. Each basin data is on a separate file, and called in by name.

Comments

None

Data Set Use

FT09F001

Variables Passed In

None

Local Variables

STA : Alphameric name of a basin (10 letters)
EBSN : "\$" Indicates type I elliptic coordinates
 "+" Indicates type II elliptic coordinates
 otherwise polar coordinates
DOLLAR : "\$" used for island, otherwise for regular basin
AZMTH : Slant of X-Axis from north
ALTO ALNO : Latitude/Longitude of basin center or origin
DEGREE : Angle between two rays of a polar grid
IRB IRE : Total grid points in -+P direction from tangent PT
NDB NDE : Total grid points in -+Q direction from tangent PT
XMOUTH : X-Coordinate from tangent point to (X,Y)=(P,Q)=(0,
YMOUTH : Y-Coordinate from tangent point to (X,Y)=(P,Q)=(0,
PHI : Slant from X-AXIT to N/S-AXIS
IMXB JMXB : Total grid spacing in P/Q directions
IMXB1 : IMXB less 1
JMXB1 : JMXB less 1

Subroutines Calling Function (File:Subroutine):

runslhg.f: INITLZ

Subroutine Calls (File:Subroutines)

None

Return

None

TTIMER:

Author

Jelesnianski/Chen, July 1989 TDL/MDL

Purpose

Keeps track of time and accumulates time steps. Function also updates the storm position at each time step on a Cartesian grid and computes growth factor to update a storm to maturity.

Comments

This subroutine resides in the root overlay

Data Set Use

None

Variables Passed In

None

Local Variables

IC19 : Storm growth time to maturity, set in *INTVAL*

ITIME : Increases by one for each time step

BT : Storm growth coefficient, $0 < BT \leq 1$

Subroutines Calling Function (File:Subroutine):

runslhg.f: CMPUTE,

Subroutine Calls (File:Subroutines)

None

Return

None

CMPUTE:**Author**

Jelesnianski, September 1980 MDL/TDL

Updated: Sung Kim, January 1992

To save data, press "P"

To pause run, press "S"

Saved data: integer*2 of $1/10^{\text{th}}$ of surge height

10000^{th} power of transport normalized by depth

Purpose

This 2nd overlay CMPUTE uses subroutine CMPUTE as a direct calling sequence for 12 subroutines. The purpose of the overlay is to compute surge heights using algorithms for partial differential equations of motion. Surges are stored for later print out.

Comments

This subroutine resides in overlay CMPUTE.

Data Set Use

None

Variables Passed In

None

Local Variables

ITIME : Cumulative time counter
 MHALT : Halt computations at this time step
 NPSS : Number of channels
 NCUT : Number of cuts along barriers
 NSQRWC : Number of squares for 1D flow plus flow cuts
 NDLTHR : Number of time steps in one hour
 IPRTSV : Time to save historical surges, 10 points, set *INTVAL*
 IPRTAD : Update time of IPRTSV, set in *INTVAL*

Subroutines Calling Function (File:Subroutine):

runslhg.f: MAINE

Subroutine Calls (File:Subroutines)

runslhg.f: SETCMP, INTLHT, RDLTLG, TIMER, CONTTY, SMPT2G,
 BDRYHT, FLW1DM, HMXSV, HISTRY, HISTRY2,
 FLTER2, FILTER, MOMNTM, STMVAL FILTRP,
 FHMXXSV, ARCHIV

Return

None

INITLZ:**Author**

Jelesnianski, September 1980 MDL/TDL

Purpose

This second overlay INITLZ uses subroutine INITLZ as a calling sequence for 5 subroutines. The purpose of the overlay is to fix some program constants, some polar grid constants; to set bathy/topo and barrier heights, and to initialize a timing sequence.

Comments

This subroutine resides in overlay INITLZ

Data Set Use

None

Variables Passed In

None

Local Variables

ALTO(DEG) : Latitude of basin center or origin

Subroutines Calling Function (File:Subroutine):

runslhg.f: MAINE

interface.f: INITAL

Subroutine Calls (File:Subroutines)

runslhg.f: INPUTN, CRDRD1, STERON, CRDRD2, DEPMSY,
DEPSFC, BSNPRT, BTMSTR, SETDST, INTDLT, INTVAL,
CNSTNT

Return

None

INPUTN:**Author**

Unknown

Purpose

Unknown

Comments

None

Data Set Use

None

Variables Passed In

None

Local Variables

None

Subroutines Calling Function (File:Subroutine):

runslhg.f: INITLZ

Subroutine Calls (File:Subroutines)

None

Return

None

STERON:**Author**

Jelesnianski/Chen, October 1989 MDL/TDL

Purpose

This subroutine changes hourly storm track positions (latitude and longitude) to positions on (X,Y) plane and records hourly speeds and directions in (X,Y) plane.

Comments

None

Data Set Use

None

Variables Passed In

None

Local Variables

XLAT() YLONG() : LAT/LON of hourly storm positions
 X() Y() : Components of hourly storm positions
 ALTO ALNO : LAT/LON of basin origin
 PHI : Slant from N/S axis to Y axis; EG;
 "AZMTH+270"
 ERAD : Radius of earth (Spherical)
 FTPNM : No of fee in a nautical mile
 XMOUTH YMOUTH: Mouth coordinates in X-Y plane

Subroutines Calling Function (File:Subroutine):

runslhg.f: INITLZ

Subroutine Calls (File:Subroutines)

None

Return

None

SETDST:

Author

Jelesnianski, September 1980 MDL/TDL

Purpose

This subroutine computes size specifics of the polar grid, as well as the orientation of the (P,Q) polar system versus the (X,Y) system. The end results are used for driving forces transformed from the (X,Y) to (P,Q) systems. Print out of polar specifics are effected.

Comments

Subroutine resides in overlay *INITLZ*.

Data Set Use

None

Variables Passed In

None

Local Variables

DELA	: Radian measure between contiguous rays
IMXB JMXB	: Total grid spacing, set in <i>CRDRD1</i>
IRB JRB	: Total grid points in +-DIRECTN from origin
NDB NDE	: Total grid points in +-Q direction from origin
RMOUTH	: Distance from pole to basin origin in miles
SIDEHZ()	: Arch length between contiguous grid points in feet
ARESQ()	: Area of squares, between concentric circles in FT- 2
SIDEVT()	: Distance between concentric circles in miles and feet.
DEGREE	: Angle between 2 contiguous rays
ANGD()	: Angle between P-AXIS and any ray, in degrees
RAD	: Radian measure
COST() SINT()	: Ray projections through momentum points, on (x,y) axis
COSL() SINL()	: Ray projections through surge points, on (x,y) axis

Subroutines Calling Function (File:Subroutine):

runslhg.f: INITLZ

Subroutine Calls (File:Subroutines)

None

Return

None

DEPMSY:

Author

Unknown

Purpose

Unknown

Comments

None

Data Set Use

None

Variables Passed In

None

Local Variables

None

Subroutines Calling Function (File:Subroutine):

runslhg.f: INITLZ

Subroutine Calls (File:Subroutines)

None

Return

None

INTLHT:

Author

Jelesnianski, September 1980 MDL/TDL

Purpose

Initializes the sea with quiescent water levels plus static heights with the storm in deep water. Static heights are not added to inland water bodies and inland water heights are initialized in subroutine *DEPTHB*

Comments

This subroutine resides in overlay CMPUTE.

Data Set Use

None

Variables Passed In

None

Local Variables

IMXB1 JMXB1	: Max subscripts for HT points
MF()	: Momentum P subscript, for end of lake winds
AX AY	: Components of storm traverse for 1 st unit of time
C1, C2	: Components of storm position, time=0, set in INTVAL
COSL() SINL()	: Components of azimuth on height points
GRIDRH()	: RADII of concentric circles through height points
SEADTM	: initial, quiescent, sea surface height
DTMLAK	: Initial water height (interior water bodies)
DELP(800)	: Static heights at mile intervals from storm center
HB(,)	: Surge heights
UB VB(,)	: Transport components
HMX(,)	: Max surge heights
ZB(,)	: Depth values on two dimensional stair steps

Subroutines Calling Function (File:Subroutine):

runslhg.f: CMPUTE

interface.f: INITAL

Subroutine Calls (File:Subroutines)

None

Return

None

RDLTLG:

Author

Unknown

Purpose

Unknown

Comments

None

Data Set Use

None

Variables Passed In

None

Local Variables

None

Subroutines Calling Function (File:Subroutine):

runslhg.f: CMPUTE

interface.f: INITAL

Subroutine Calls (File:Subroutines)

None

Return

None

CRDRD2:

Author

Unknown

Purpose

Read boundary types along left, right, top, and bottom boundaries along momentum points

Comments

None

Data Set Use

None

Variables Passed In

None

Local Variables

None

Subroutines Calling Function (File:Subroutine):

runslhg.f: INITLZ

Subroutine Calls (File:Subroutines)

None

Return

None

DEPSFO:

Author

Jelesnianski, September 1980 MDL/TDL

Purpose

Initializes inland surface waters to lake datum (sea water to sea datum is done in subroutine *INILHT*). Barriers and their heights are placed at selected momentum grid points. For 1-DIM flow, depths and lateral boundaries are set. Boundaries are temporarily set as a 100 foot wall. Print outs are made of the depth field, and highest land value at momentum grid points are set.

Comments

Subroutine resides in overlay *INITLZ*.

Data Set Use

None

Variables Passed In

None

Local Variables

IMXB JMXB	: Total grid spacing, set in <i>CRDRD1</i>
IMXB1 JMXB1	: IMXB less 1, JMXB less 1
DOLLAR	: \$ for island, otherwise for regular basin

ZB(,)	: Bathymetry/topographic heights, relative to NGVD
ZBM(,)	: Highest elevation at a momentum point or values indicating boundary types
NSQRW	: Number of squares with 1-DIM flow
ISQR JSQR()	: Subscripts for squares with 1-DIM flow
IIH(4) JJH(4)	: Shift subscripts to 4 momentum points on a square
HWEIR()	: Lowest barrier elevation at ISIDE()
IZ1(4) IZ2(4)	: Set P subscripts on 4 sides of ISIDE()
JZ1(4) JZ2(4)	: Set Q subscripts on 4 sides of ISIDE()
ZBMIN()	: Lowest depth of two squares bounding ISIDE(30)
MGPT	: Total number of mangrove points
IMG() JMG()	: I/J coordinates of mangrove points
ITREE (,)	: 0 for no tree, 1 for tree on a grid point

Subroutines Calling Function (File:Subroutine):

runslhg.f: INITLZ

Subroutine Calls (File:Subroutines)

None

Return

None

CONTTY:

Author

Jelesnianski, September 1980 MDL/TDL

Purpose

This subroutine is an algorithm for the continuity equation, for an image plane; polar coordinates are transformed onto the image plane. The system axis are (P,Q), but the subroutine uses (I,J) for integer subscripting two levels in time (forward in *CONTTY* plus backward in MOMN) are used. The grid (See Diagram below is staggered in space a 5-point scheme (4 corners of a square) are used for the momentum space gradient.

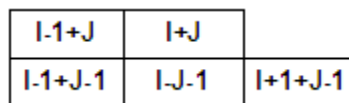


Figure 21 – 5 Point Grid Scheme

When water descends below square (I,J), the momentum on 4 surrounding corners is reduced to allow square to become dry. The previously computed surges on the 4-surrounding squares are updated according to the revised momentums on the 4-corner points of (I,J)

Comments

- 1) This subroutine resides in overlay *CMPUTE*.
- 2) The system routine *ERRSET* is used to isolate variables if an under/overflow occurs; Variable printout is aided via *HELP* and the *COMMON/CHELP/*
- 3) The subscript for computations of surges are offset (.5, .5) from the momentum points – see below.

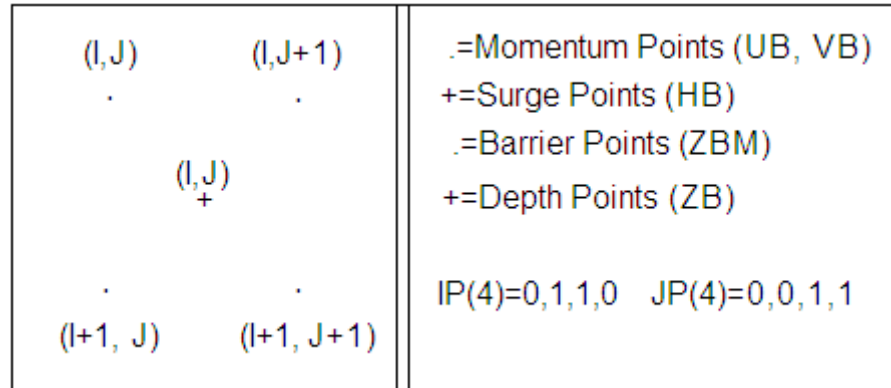


Figure 22 – Surge offset Computations From Momentum

Data Set Use

None

Variables Passed In

None

Local Variables

IMXB1 JMXB1	: Max subscripts for surge squares
HB(,)	: Surge heights
HSUB(,)	: Storage for a second copy of HB
IS()	: Start loop, exclude high terrain, set in CRDRD2
IB IFN	: Beginning/Ending a “Do Loop” in I-direction
ISKIP	: 1, for J=1, check ZBM at 2 boundary corners for skipping static condition 2, and I=IMXB1 for bottom boundary 3, for J=JMXB1 for right boundary
JUMP	: Conditional integer for standard loop or retrieval of previously computed surges, see diagram above for affected squares
LEAP	: Conditional “GO TO” to dry an (I,J) square, see diagram above for an (I,J) square
IFST ISND	: Begin/End I subscript on updated (J-1) squares
IPASS	: Conditional integer, to set a square dry
ZB(,)	: Depths at center of surge squares
IP(4) JP(4)	: Shift subscripts to 4 momentum points
UB VB(,)	: Components of momentum fields

HPST	: Dummy value for surge values
FCT	: Factor to multiply momentum gradient
CHNG	: Surge plus depths (to test for dry square)
DENM	: Change in surge, on the time iteration
ANUM	: Change in surge ht, to dry a square
RFCT	: Ratio, < 1, of ANUM to DENM, reduction factor
ISET	: I-subscript for dried square

Subroutines Calling Function (File:Subroutine):

runslhg.f: CMPUTE

interface.f: TMSTEP

Subroutine Calls (File:Subroutines)

None

Return

None

BDRYHT:

Author

Jelesnianski, September 1980 MDL/TDL

Purpose

Computes static heights on deep water boundaries only. Deep water can be on one, tow or neither of the two side boundaries, but it will exist at least on a segment of the circle with largest radius of the polar grid.

Comments

1) This subroutine resides in overlay CMPUTE

Data Set Use

None

Variables Passed In

None

Local Variables

AX AY	: Comps of total storm motion, advanced in <i>STMVAL</i>
C1 C2	: Initial components storm, set in <i>INTVAL</i>
COSL() SINL()	: CO-SINE of angle, rays to X-AXIS, (Height points
IMXB1	: Max I-Subscript for height points
SEADTM	: Initial height of the sea (no static heights)
DELP(800)	: Static heights at mile intervals from storm center
HB (,)	: Surge heights

ITREE(,) : A, on at least one boundary corner as static height is used

Subroutines Calling Function (File:Subroutine):

runslhg.f: CMPUTE

interface.f: TMSTEP

Subroutine Calls (File:Subroutines)

None

Return

None

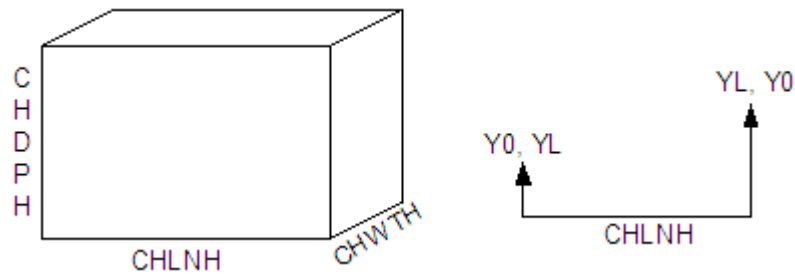
CHANNL:

Author

Jelesnianski, September 1980 MDL/TDL

Purpose

Computes total flow through a prism, invariant in space. The prism bed is horizontal. The head between the two ends is the driving force to pass water through the channel. See image below:



The equation used for the channel flow is:

$$Q^{**2} = (C1(YL^{**13/3} - Y0^{**13/3})) / (C2(YL^{**4/3} - Y0^{**4/3}) + CHLNH)$$

Figure 23 – Channel Flow Equation

Comments

This subroutine resides in overlay *CMPUTE*

Data Set Use

None

Variables Passed In

None

Local Variables

NPSS : Number of passes

CHLNH(5) : Length of channels

G	: Gravity
Y0 YL	: Surge at end of channel
IPTO JPTO(2,5)	: Positions for 2-squares at one channel end
HB(,)	: Surge height
CHDPH(5)	: Channel depths
IPTL JPTL(2,5)	: Positions of 2-squares at other channel end
JPTL(2,5)	: J-Subscript for 2-squares at other channel end
SIG	: Sign , + or –
CDELY	: Cut-off height of .1 foot for the head
AMAN-AGB13	: Fixed constants
Q(5)	: Flow
DELT	: Unit time step
CHWTH(5)	: Width of channels
EXTXT(5)	: Total flow through channel

Subroutines Calling Function (File:Subroutine):

runslhg.f: CMPUTE

interface.f: TMSTEP

Subroutine Calls (File:Subroutines)

None

Return

None

FLW1DM:

Author

Jelesnianski, September 1980 MDL/TDL

Purpose

This subroutine computes 1-dimension flow across channel sides of interlocking squares. Water passes only through OPEN sides of a 1-dimension flow channel, see diagram below.

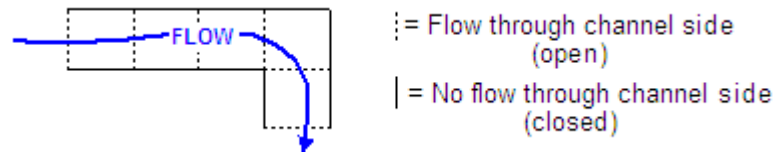


Figure 24 – Computed 1-Dimension Flow Across Channel of Interlocking Squares

The *loop* of the continuity equation fills/depletes two adjoining squares with flow through an *open* side, one side at a time. If water drops below a square from the action of an *open* side, then the flow through the side is reduced to exhaust water only to a dry square.

Each open side has an interior and an exterior square. Subroutine *CRDRD2* reads the subscripts for interior squares and sets up procedures

to isolate the adjoining exterior squares and open sides via *COMMON/DUMB18/*. The present subroutine isolates the two interior/exterior squares and the adjoining side for 1-dimension flow. A 1-dimension flow square can be either an int/ext square, or both, 1-4 times, depending on the amount of open sides bounding a square

I,J shifts about an (I,J) interior square for 4 possible exterior squares which are as shown:

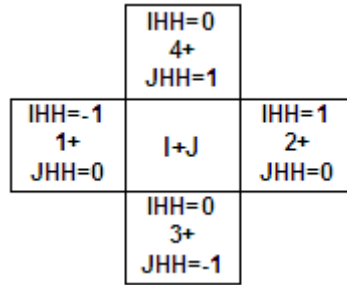


Figure 25 – Interior/Exterior Calculated Flow For Individual Grid Cell

Comments

- 1) This is a version for 2-step scheme (July 1981).
- 2) Subroutine resides in overlay CMPUTE.

The diagram below shows some (not all) elevations on interior/exterior squares at present time:

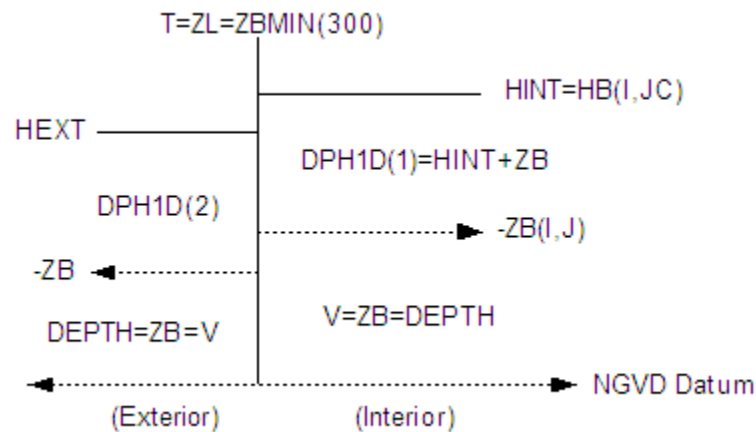


Figure 26 – Elevation on Interior/Exterior Calculated Flow For Individual Grid Cell

Data Set Use

None

Variables Passed In

None

Local Variables

AI(800)	: Bottom friction, at one foot intervals
NSQRW	: Number of squares with 1-dimension flow possible
ISQR*L) JSQRL(L)	: I/J-subscript for square with 1-dimension flow
HB(,)	: Surge field
ISIDE(300)	: Side of flow relative to interior 1-dimension flow square
IHH(4) JHH(4)	: Shift I/J-subscript for contiguous square
HINT HEXT	: Interior/Exterior surge for surface gradient
HINT1 HEXT1	: Interior/Exterior surge heights on squares
ZBMIN(300)	: Lowest barrier value at end of an open side
ZL	: Dummy value for ZBMIN(300)
ZB(,)	: Depth field
HWEIR(300)	: highest depth of Interior/Exterior squares or WEIR height
HOVT	: Dummy value for HWEIR(300)
FLWSQR(300)	: Transport across an open side, two time levels
GRIDR2()	: Jacobean for polar coordinates
SIGA(4)	: Sign for flow across sides of squares
DPH1D	: Depth + Surge at interior/exterior squares, present time
CHNGHI CHGHE	: Surge increase on interior/exterior squares from open side
DEPLI DEPLE	: Total depth of interior/exterior squares, future time
CII	: Accumulated water from open sides, interior square
SUBE	: Surge from external square to prevent water below interior square
CEE	: Accumulated water from open sides, exterior square
ID DPH	: Mean total depth of interior/exterior squares, present time
DZL	: Height of HWEIR above mean depths
FXB FYB	: Components of surface stress
X1B(N)	: Fixed coefficients for finite differencing
BR(300) BI(300)	: Friction coefficients at 1-foot intervals
JUMP	: 2 for stress computations, 1 for no stress

Subroutines Calling Function (File:Subroutine):

runslhg.f: CMPUTE

interface.f: TMSTEP

Subroutine Calls (File:Subroutines)

None

Return

None

FRCPNT:

Author

Jelesnianski, September 1980 MDL/TDL

Purpose

Computes components of surface driving forces at a momentum grid point. Entry values for computations are through COMMON/MF/. The lake winds are distorted according to forward speed along track; Ocean winds are not distorted.

Comments

1) Subroutine lies in overlay CMPUTE

Data Set Use

None

Variables Passed In

X	: See below
Y	: TBD
FXB	: Components of surface forces on image plane
FYB	: Components of surface forces on image plane
NCATG	: TBD
CHLSTR	: Extinction coefficient of stress for water less than 1 ft.
NPLS	: TBD

Local Variables

AX AY	: Components of total storm traverse
C1 C2	: Initial (X,Y) storm position, set in INTVAL
X12(4)	: Radius of max wind in feet
X12(5)	: Square of X12(4)
X12(9) X12(10)	: Components of forward speed, 1 st hour
C19	: Friction coefficient (3x10 ⁻⁶)
W1218(2)	: Twice the maximum winds, set in CMPUTE
CW(800) SW(800)	: Inflow angle, mile increment, lake winds
C(800) S(800)	: Inflow angle, mile increment, ocean winds
BI BR CI CR(800)	: Friction factions
P(800)	: Pressure gradient, mile intervals
FXX FYY	: Components of stress
PXX PYY	: Components of surface pressure gradient force

FXBP FYBP : Components of surface driving forces in (X,Y)
 FXB FYB : Components of surface forces on image plane

Subroutines Calling Function (File:Subroutine):

runslhg.f: MOMNTM, FLW1DM, MNTMBD

Subroutine Calls (File:Subroutines)

None

Return

None

MOMNTM:

Author

Jelesnianski/Chen, September 1980 MDL/TDL

Purpose

Computes future momentum components on all interior grid points, after first computing future surge by subroutine CONTTY. It checks if MOMNTM can exist at a grid point; EG., Overtopping the grid point. . Future surges updated from subroutine CONTTY are used to compute height gradients and to check for eligible momentum points---backward scheme--- Checks are made if momentum is to be computed at open boundary points.

Comments

1) Subroutine resides in overlay CMPUTE

Data Set Use

None

Variables Passed In

None

Local Variables

NCATG : 1 for lake winds, 2 for ocean winds
 NPLS : 0 Remove static pressure term in forcing(INT.B.C.), otherwise 1.
 JMXB1 : Max number of j-points, less a boundary point
 JUMP : 1, fully flooded; 2, partially flooded (no stress)
 MS() : Begin I-subscript on a J-line
 ITREE(,) : Indicator array for lake/ocean, tree/no tree A, B, C, D for boundary types.
 ZBM(,) : Maximum barrier HT at a surge points to 4 MOM. Corners
 IH(4) JH(4) : Shift I/J-subscripts to 4 surrounding surge points

IP(4) JH(4)	: Shift I/J-subscripts from surge points to 4 MOM. Corners
HB(,)	: Surge heights
ZB(,)	: Depths
HF(4)	: Storage for 4 surrounding surge points
HPD(4)	: Storage for 4 surrounding total depth points
UB VB(,)	: Components of transport
DHX DHY	: Components of surface gradient
ID DPH	: Mean total depth
AI(300) AR(300)	: Friction coefficients on transports
FXB FYB	: Components of driving forces
X1B(4)	: DEL*CORIOLIS parameters
CSHLTR	: Extinction coefficient of stress for water less than 1 foot
ISKIP	: 1, compute momentum on J=1 boundary 2, compute momentum on I=2 boundary 3, compute momentum on J=JMXB boundary 4, compute momentum on I=IMXB boundary
IJUMP	: 1, shallow water boundary computation
ISBS JSBS(4)	: Shift subscripts from interior to boundary point MOM-points for all 4 corners
ISHF(3,4)	: Subscripts shift from interior to 3 corner MOM-points for all 4 corners

Subroutines Calling Function (File:Subroutine):

runslhg.f: CMPUTE

interface.f: TMSTEP

Subroutine Calls (File:Subroutines)

runslhg.f: MNTMBD, FRCPNT

Return

None

MNTMBD:

Author

Jelesnianski, October 1980 MDL/TDL

Purpose

Computes transports on momentum boundary points, between land and deep waters. In shallow waters the boundary conditions equals surface gradient at boundary and nearest interior point. In intermediate water the boundary conditions ignores time variations and equates surge gradient as static height gradient at a boundary point. Below is arrangement of momentum boundary line and nearest interior line.

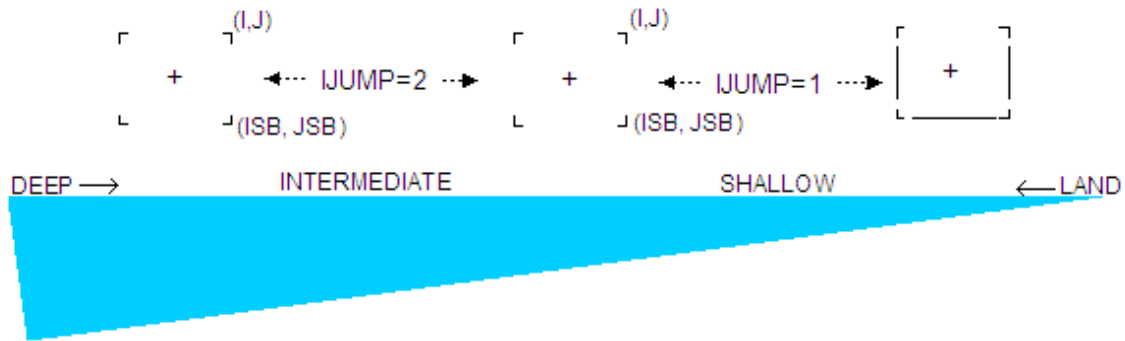


Figure 27 – Momentum Boundary Between Land and Deep Water

Comments

1) Subroutine resides in overlay *CMPUTE*.

Data Set Use

None

Variables Passed In

JUMP	: 1 FOR surface forcing, 2 for no surface forcing
IJUMP	: 1 for shallow water boundary computations 2 for interior depth boundary computations
ISB	: Revised subscripts for I boundary point
JSB	: Revised subscripts for J boundary point
ARJ	: Pre-computed values in MOMNTM
AII	: TBD
I	: TBD
J	: TBD
UBJ VBJ	: Transports at (I,J) at present time, saved before updated
VBJ	: Transports
FXB	: Component of driving force
FYB	: Component of driving force
NCATG	: TBD
CSHLTR	: TBD
NPLS	: TBD

Local Variables

ISKIP	: 1, to compute momentum on J=1 boundary 2, to compute momentum on I=IMXB boundary 3, to compute momentum on J=JMXB boundary
AIT AII AIU	: Pre-computed values on MOMNTM
AIV, XTT	: Pre-computed values on MOMNTM
AINTRP	: Pre-computed values on MOMNTM
X1B(10)	: Fixed constants

Subroutines Calling Function (File:Subroutine):

runslhg.f: MOMNTM

Subroutine Calls (File:Subroutines)

runslhg.f: FRCPNT

Return

None

FHMCSV:

Author

Unknown

Purpose

Unknown

Comments

None

Data Set Use

None

Variables Passed In

None

Local Variables

None

Subroutines Calling Function (File:Subroutine):

runslhg.f: CMPUTE

interface.f: CLNUP

Subroutine Calls (File:Subroutines)

None

Return

None

ARCHIV:

Author

Unknown

Purpose

Read boundary types along left, right, top, and bottom boundaries along momentum points

Comments

None

Data Set Use

None

Variables Passed In

None

Local Variables

None

Subroutines Calling Function (File:Subroutine):

runslhg.f: CMPUTE

interface.f: CLNUP

Subroutine Calls (File:Subroutines)

None

Return

None

HISTORY:

Author

Jelesnianski/Chen, September 1980 MDL/TDL

Purpose

Smooths the surface at 10 pre-selected grid points and saves them on disk for future printout. Also, wind speed and direction are also saved at the same 10 points.

Comments

1) Subroutine resides in overlay *CMPUTE*.

Data Set Use

FT 10, temporary scratch disk for surge storage

FT 20, temporary scratch disk for surge storage

FT 30, temporary scratch disk for surge storage

Variables Passed In

None

Local Variables

IPN(10) IPL(10) : Subscripts for 10 selected points

HB(,) : Surge field

ZB(,) : Depth field

IP(4) JP(4) : Shift subscripts to 4 adjacent momentum points

ZBM(,) : Maximum barrier height at a momentum point

COST() SINT() : Ray projections through MOMNTM points, on (X,Y) axis

GRIDRF() : Radial distances to momentum points

AX AY	: Components of total storm traverse, advanced in STMVAL
C1 C2	: Components of initial storm position
NCATG	: 1 for lake winds, 2 for ocean winds
W1218(2)	: Twice the maximum winds, land/ocean winds
CW(800) SW(800)	: Inflow angle for lake winds
C(800) S(800)	: Inflow angle for ocean winds
X12(50)	: Fixed coefficients
X12(9) X12(10)	: Components of forward speed, 1 st hour
WIND(10)	: Wind speed at 10 selected points
TDIR(10)	: Wind direction at 10 selected points
PHI	: Slant, Y-Axis to N/S direction, AZMTH+270
RAD	: Radian Measure
IPRHR	: 1, when on the hour; 2, when half hour in between; 3, otherwise

Subroutines Calling Function (File:Subroutine):

runslhg.f: CMPUTE

interface.f: TMSTEP

Subroutine Calls (File:Subroutines)

None

Return

None

HSTRY2:

Author

Unknown

Purpose

Modified SLOSH HSTRY function for evacuation study.

Comments

See HSTRY subroutine

Data Set Use

See HSTRY subroutine

Variables Passed In

See HSTRY subroutine

Local Variables

See HSTRY subroutine

Subroutines Calling Function (File:Subroutine):

runslhg.f: CMPUTE

interface.f: TMSTEP

Subroutine Calls (File:Subroutines)

None

Return

None

FILTRP:

Author

Jelesnianski, October 1980, MDL

Purpose

Smooths the computed surge envelope of highest waters, in space. The procedures are the sillier in FILTER and HMXSV, 2 passes of 3-point smoother. It is for cosmetic appearance. No smoothing across the non-overtopped barriers is allowed.

Comments

Subroutine resides in overlay *PLOTS*.

Data Set Use

None

Variables Passed In

None

Local Variables

IMAX JMAX	: Maximum grid points in the basin
IMXB1 JMXB1	: IMAX less 1, JMXB less 1
HMX(,)	: Surge envelope
HZ(, 3)	: Dummy storage for 2 lines
ZB(,)	: Depth field
IIH(4) JJH(4)	: Shift subscripts to 4 adjoining momentum points
IP(4) JP(4)	: Shift subscripts to 4 adjoining surge points
ZBM(,)	: Max barrier heights at momentum points

Subroutines Calling Function (File:Subroutine):

runslhg.f: CMPUTE

interface.f: CLNUP

Subroutine Calls (File:Subroutines)

None

Return

None

HMXSV:

Author

Jye Chen, September 1980, MDL

Purpose

Stores maximum surge as advancing in time at each grid point, into array HMX(,). The smoothed surge value is computed and compared locally against previously saved HMX. The reason is to avoid temporary noise entering the final envelop.

Comments

- 1) Subroutine resides in overlay *CMPUTE*.
- 2) A 5-point smoothing operator similar to one in subroutine FILTER is used, see FILTER.

Data Set Use

None

Variables Passed In

None

Local Variables

HMX(,)	: Maximum surge at each grid point
IP(4) JP(4)	: Shifts of I J from height point to 4 MOMNTN corners
IBB(6, M)	: Ranges and increments for 3 do-loops of I,J M=1 interior, M=2 Horizontal body M=3 vertical boundary ranges on three basin segments are – see FILTER
HB(,)	: Surge field
ZB(,)	: Depth field
IIH(4) JJH(4)	: I/J- Shift subscripts for surge points

Subroutines Calling Function (File:Subroutine):

runslhg.f: CMPUTE

interface.f: TMSTEP

Subroutine Calls (File:Subroutines)

None

Return

None

SMPT2G:

Author

Jye Chen, August 22, 1984, MDL

Purpose

To collect momentum grid points adjacent to any interior boundaries (U=V=0) then convert to height points. Repetition is avoided. Special points, one of TH 4 is dry, are tagged as 888. For heavier smoothing. Call subroutine FLT2G for smoothing.

Comments

None

Data Set Use

None

Variables Passed In

None

Local Variables

HSUB(,) : Temporary storage space for height field

Subroutines Calling Function (File:Subroutine):

runslhg.f: CMPUTE

interface.f: TMSTEP

Subroutine Calls (File:Subroutines)

runslhg.f: FLT2G

Return

None

FLT2G:

Author

Jye Chen, August 22, 1984, MDL

Purpose

To eliminate 2-grid computational noise nearby land-water boundaries. Grid points nearby interior boundaries are picked out from testing in subroutine SMP2G. Smoothing does conserve mass by assuming outside the selected points do not interact with the points to be smoothed.

Comments

None

Data Set Use

None

Variables Passed In

None

Local Variables

MM2G	: Total number of points to be smoothed
MCMX	: Special smoothing points
I2G() J2G()	: I, J Coordinates of grid points for smoothing
HB(,)	: Surge height field
ZB(,)	: Terrain heights
ZBM(,)	: Barrier heights
HCT	: Cutoff depth, 1 ft except the special points which 0 ft is used

Subroutines Calling Function (File:Subroutine):

runslhg.f: SMPT2G

Subroutine Calls (File:Subroutines)

None

Return

None

FILTER:

Author

Jye Chen, September 1980, MDL

Purpose

To smooth the surge heights once an hour to eliminate 2-interval noises in the field. The (I,J) shifts to momentum points, VIA IP(4) and JP(4) on corner points k=1 to 4 are:

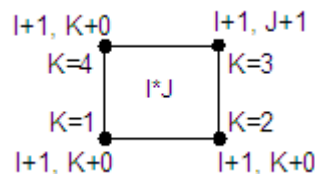


Figure 28 – Single Grid Cell Smoothing (Filter Function)

Comments

- 1) This subroutine resides in overlay *CMPUTE*.
- 2) A 5-point smoothing operator is used, neighboring grids weighted conditionally if they are dry or blocked by barriers.
- 3) Shifts of (I, J) to adjacent squares are set VIA IIH(4) and JJH(4).
IIH/0, 1, 0, -1/, JJH/-1, 0, 1, 0/

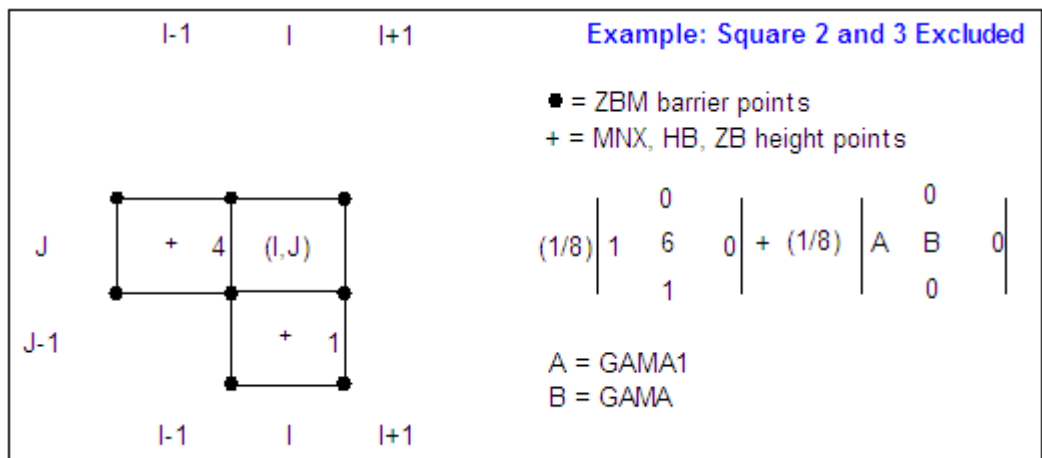
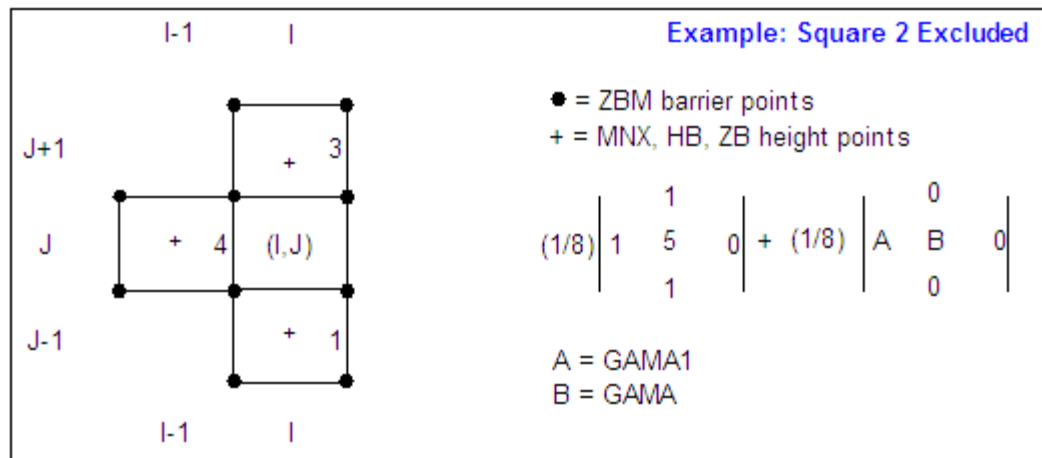
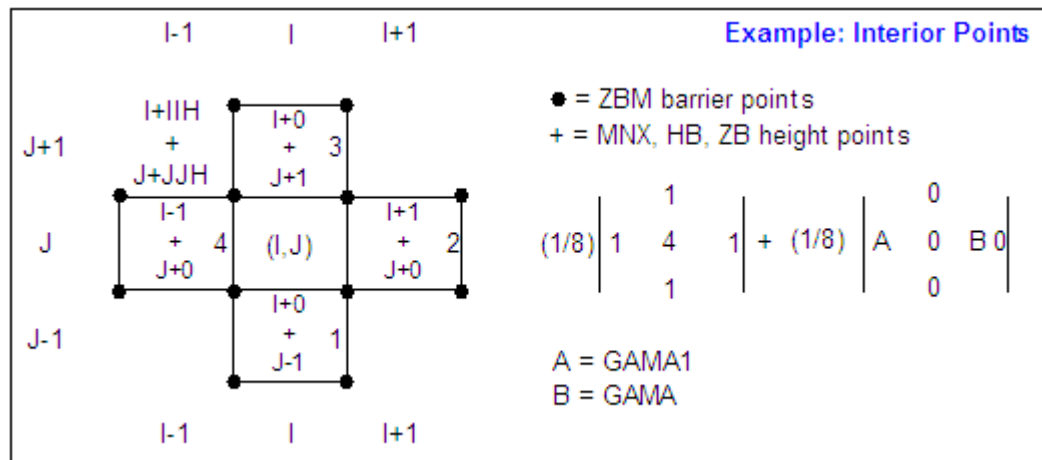


Figure 29 – Multiple Grid Cell Smoothing (Filter Function)

Data Set Use

None

Variables Passed In

None

Local Variables

HSUB(,) : Scratch spaces
 IP(4) : Shifts of I J from height point to 4 MOMNTN corners
 IBB(6,M) : Ranges and increments for 3 do-loops of I, J.
 M=1 interior, M=2 horizontal body, M=3 vertical boundary ranges on three basin segments

```

222222222.....22222222
311111111.....11111113
311111111.....11111113
.                  . J
.                  . J
311111111.....11111113 J
311111111.....11111113 J
222222222.....22222222 → I

```

J2(J) J(C) : Shift J-subscript to present time of surge field
 HB(,) : Surge field
 ZB(,) : Depth field
 IIH(4) JJH(4) : I/J-shift subscripts for surge points
 IP(4) JP(4) : I/J-shift subscripts for MOMNTM points in BLCKDT
 ZBM(,) : Max barrier heights at momentum points

Subroutines Calling Function (File:Subroutine):

runslhg.f: CMPUTE
 interface.f: TMSTEP

Subroutine Calls (File:Subroutines)

None

Return

None

FLTER2:

Author

Unknown

Purpose

Unknown

Comments

None

Data Set Use

None

Variables Passed In

None

Local Variables

TBD

Subroutines Calling Function (File:Subroutine):

runslhg.f: CMPUTE

interface.f: TMSTEP

Subroutine Calls (File:Subroutines)

None

Return

None

SNAPSH:**Author**

Unknown

Purpose

Convert surge envelop data with 999 for dry squares so that ZB() can be free from memory

Comments

None

Data Set Use

None

Variables Passed In

None

Local Variables

TBD

Subroutines Calling Function (File:Subroutine):

runslhg.f: STMVAL

Subroutine Calls (File:Subroutines)

None

Return

None

3.3.2**Source file “smooth.f” Sub-function Breakdown**

FILT3:

Author

- 1) Jye Chen, September 1980, MDL/TDL FILTER in “runslhg.f” (see section 2.3.1)
- 2) Modified: Arthur Taylor, Date Missing, MDL/TDL

Purpose

To smooth the surge heights once an hour to eliminate 2-interval noise in field

Comments

None

Data Set Use

None

Variables Passed In

None

Local Variables

- HSUB(,) : Scratch spaces
IP(4) : Shifts of I J from height point to 4 MOMNTN corners
IBB(6,M) : Ranges and increments for 3 do-loops of I, J.
M=1 interior, M=2 horizontal body, M=3 vertical boundary ranges on three basin segments

```
22222222.....22222222
31111111.....11111113
31111111.....11111113
.                  . J
.                  . J
31111111.....11111113 J
31111111.....11111113 J
22222222.....22222222 → I
```

- J2(J) J(C) : Shift J-subscript to present time of surge field
HB(,) : Surge field
ZB(,) : Depth field
IIH(4) JJH(4) : I/J-shift subscripts for surge points
IP(4) JP(4) : I/J-shift subscripts for MOMNTM points in BLCKDT
ZBM(,) : Max barrier heights at momentum points

The (I,J) shifts to momentum points, VIA IP(4) and JP(4) on corner points k=1 to 4 are:

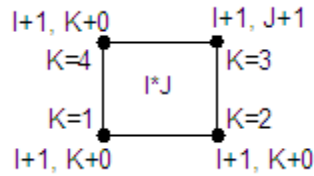


Figure 30 – Single Grid Cell Smoothing (Smoothing Function)

Comments

- 1) This subroutine resides in overlay CMPUTE.
- 2) A 5-point smoothing operator is used, neighboring grids weighted conditionally if they are dry or blocked by barriers.
- 3) Shifts of (I, J) to adjacent squares are set VIA IJH(4) and JJH(4).
IJH/0, 1, 0, -1/, JJH/-1, 0, 1, 0/

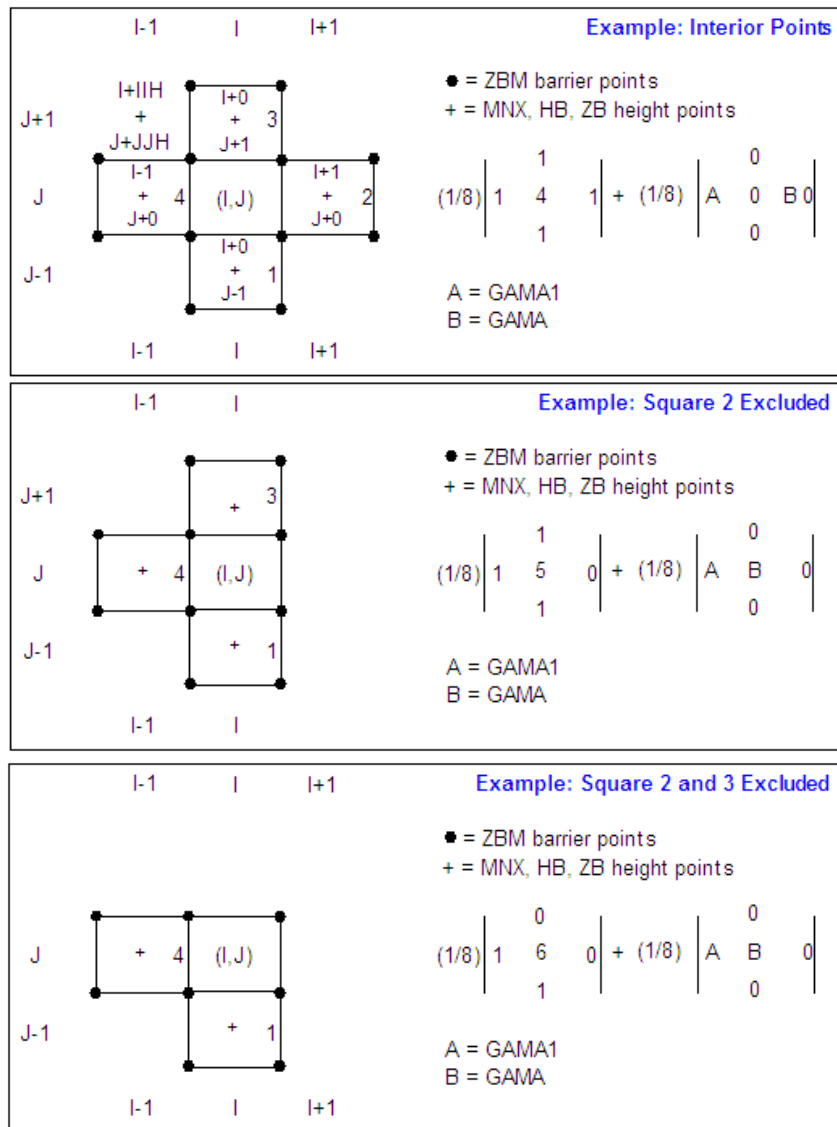


Figure 31 – Multiple Grid Cell Smoothing (Smoothing Function)

Subroutines Calling Function (File:Subroutine):
interface.f: TMSTEP

Subroutine Calls (File:Subroutines)
None

Return
HHB

FLTER4: **Author**
Unknown

Modified: Arthur Taylor, Date Unknown MDL

Purpose
Originally filter 2 in “runslhg.f”, otherwise unknown

Comments
None

Data Set Use
None

Variables Passed In
None

Local Variables
TBD

Subroutines Calling Function (File:Subroutine):
interface.f: TMSTEP

Subroutine Calls (File:Subroutines)
None

Return
HHB

3.3.3 Source file “interface.f” Sub-function Breakdown

INITAL: **Author**
Unknown

Purpose

This subroutine calls all the FORTRAN subroutines which are needed to initialize the FORTRAN variables before we enter into the compute loop. Originally this was done in program main and CMPUT. The code is the same except cut and was pasted here. This function also returns to C the various initial variables needed so that it can properly control the execution of the program.

Comments

None

Data Set Use

None

Variables Passed In

MMHALT	:	TBD
IIMXB	:	TBD
JJMXB	:	TBD
ZZB	:	TBD
MBHR	:	TBD
MMIN	:	TBD
MBDY	:	TBD
MBNT	:	TBD
MYR	:	TBD
YYLT	:	TBD
FFLE5	:	TBD
FFLE9	:	TBD
FFLE91	:	TBD
FFLE99	:	TBD
FFLE40	:	TBD

Local Variables

TBD

Subroutines Calling Function (File:Subroutine):

Slosh2.c: RunInit

Subroutine Calls (File:Subroutines)

runslhg.f: INITLZ, INTLHT, RDLTLG, SETCMP

Return

MMHALT, IIMXB, JJMXB, ZZB, MBHR, MBDY, MBNT, MYR, YYLT, YYLG

WIND:**Author**

This procedure is from Jye Chen's wind2.for (first 3 pages). Modified by Arthur Taylor 6/9/97

Purpose

Procedure calculates the SLOSH wind grid on a lat/long grid. Results are returned in MPH.

Comments

- 1) Storm center is at center of grid and may not actually need to call force1

Data Set Use

None

Variables Passed In

IU(0)	:	MPH wind in U direction at location I, J (Max:200x200)
IV(0)	:	MPH wind in V direction at location I,J
IP(0)	:	Pressure at location I,J
DELSNM(I)	:	DELTA in nautical miles on grid (usual:5)
N0(I)	:	Number of grid elements in 2D array (max:199) (usual: 95)
WSPEED(I)	:	The storms speed in MPH
WDIRECT(I)	:	The storms direction (0 for north)

Local Variables

TBD

Subroutines Calling Function (File:Subroutine):

Slosh2.c: #define

Subroutine Calls (File:Subroutines)

runslhg.f: FORCE1

Return

MMHALT, IIMXB, JJMVB, ZZB, MBHR, MBDY, MBNT, MYR, YYLT, YYLG

WAV:

Author

Unknown

Purpose

Unknown

Comments

None

Data Set Use

None

Variables Passed In

None

Local Variables

TBD

Subroutines Calling Function (File:Subroutine):

interface.f: TMSTEP

Subroutine Calls (File:Subroutines)

None

Return

None

TMSTEP:

Author

Unknown

Purpose

This subroutine calls all the FORTRAN subroutines which are needed to do one time step. Originally this was done in CMPUT between 100 and 99. The code is the same except for cut and paste here. This also returns to C the various variables needed so that it can update the graphics.

Comments

None

Data Set Use

None

Variables Passed In

IITIME	:	TBD
MMHALT	:	TBD
HHB	:	TBD
DDELT	:	TBD
XXLT	:	TBD
XXLG	:	TBD
YYDELP	:	TBD
YYC24	:	TBD
1WSPEED	:	TBD
WDIRCT	:	TBD
FLAG	:	TBD
SMOOTH	:	TBD
FPASS	:	TBD

Local Variables

TBD

Subroutines Calling Function (File:Subroutine):

slosh2.c: RunLoopStep

Subroutine Calls (File:Subroutines)

runslhg.f: TTIMER, CONTTY, SMPT2G, BDRYHT, CHANNL,
FLW1DM, HMXSV, HISTRY HSTRY2, FLTER2, FILTER,
MOMNTM, STMVAL

interface.f: WAV, XY2LLX

smooth.f: FILT4, FILT3

Return

IITIME, MMHALT, HHB, DDELT, YYDELP, YYC24, WSPEED,
WDIRECT

CLNUP:**Author**

Unknown

Purpose

This subroutine calls the FORTRAN subroutines needed to save the max envelop and to clean up execution of the code. Originally this was done at the end of CMPUT (after 999) and the end of the program main. The code is the same except cut and pasted here. This also returns to C the various variables needed so that it can display the max envelop.

Comments

None

Data Set Use

None

Variables Passed In

HHB : TBD

iENV : TBD

Local Variables

TBD

Subroutines Calling Function (File:Subroutine):

Slosh2.c: CleanUp

Subroutine Calls (File:Subroutines)

runslhg.f: FILTRP, FHMCSV, ARCHIV

Return
HHB

LLX2PQ:

Author
Unknown

Purpose
Unknown

Comments
None

Data Set Use
None

Variables Passed In
XLTL : TBD
YLNL : TBD
INP : TBD
INQ : TBD

Local Variables
TBD

Subroutines Calling Function (File:Subroutine):
Slosh2.c: #define

Subroutine Calls (File:Subroutines)
None

Return
YLNL, INP, INQ

XY2LLX:

Author
Unknown

Purpose
Unknown

Comments
None

Data Set Use
None

Variables Passed In

XA : TBD
YA : TBD
XLT : TBD
XLG : TBD

Local Variables

TBD

Subroutines Calling Function (File:Subroutine):

interface.f: TMSTEP

Subroutine Calls (File:Subroutines)

None

Return

XLT, XLG

3.4 DOS Wrapper Source Tree

The SLOSH DOS model simply stated is C based “wrapper” software used to interact with the “base” code to execute the model from a Microsoft DOS command-line.

1. clock.c / clock.h

- Module provides various clock functions to manipulate time for SLOSH model runs. Standard libraries not robust enough.

2. complex.c / complex.h

- Module containing basic complex mathematical equations/calculation subroutines. EG, Square root, log functions etc.

3. cstart.c

- Main program, starting point for command-line (base) software

4. myassert.c / myassert.h

- Code to handle assert statements, There is no actual code unless DEBUG is defined

5. myutil.c / myutil.h

- Utility software to conduct a variety of tasks, among them: reallocating memory space, split character strings, file copies, string trimming functions, and more

6. pack.c / pack.h

- Functions to fill or remove data for selected basin. Compression algorithms for REX files and encoding/decoding REX data files

7. savellx.c / savellx.h

- Functions to support generation of llx files used by Fortran code

8. slosh2.c / slosh2.h

- Main control routines for running SLOSH

9. tendian.c / tendian.h

- Utility functions used to solve big/little endian related issues

10. tio3.c / tio3.h

- Implements binary file I/O in an endian independent manner (ie specify if the file was created on an Intel machine or not, when you open the file, and then ignore), and in such a way that it can be called from FORTRAN 77.
- If using FORTRAN generated files, they sometimes stick extra bytes at the beginning and end of a record. These are hidden from the FORTRAN user, but not from this library. They usually contain the record length. Example, in UNIX it usually stores an int*4, and in Microsoft Fortran, an int*1 with 129 being a continuation flag.

11. type.h

- Type declarations between operating systems, ie 64bit, 32 bit
- System DIFF definitions

3.4.1

Source file “clock.c / clock.h” Subroutine

Global Variables:

```
enum      : Array of time related ASCII characters
            represented in Boolean format. EG:
            WT_COLON = “:” or with (WT) colon

stackType : Struct
            val : value of int from time string
            len : length of time string in memory

relType   : Struct
            relUnit : TBD
            f_negate : TBD
            amount  : TBD
```

main:

Author

Unknown

Purpose

clock.c manipulates time to assist SLOSH model for desired time outputs.

Variables Passed In

argc & argv : Command line arguments

Local Variables

```
f_timeAdj : Numeric representation of local standard or
            local daylight time adjustments
            0 = no adjustment
            1 = adjust as LDT
            2 = adjust as LST

d_clock   : TBD
```

d_amount : TBD
 i : TBD
 TimeZone : TBD
 F_dayLight : TBD
 formatPtr : TBD
 ans : TBD
 buffer : TBD
 lenBuffer : TBD
 year : TBD
 month : TBD
 day : TBD
 totDay : TBD
 d_remain : TBD
 d_remain : TBD
 f_isyear : TBD

Subroutine Calls (File:Subroutines)

myUtil.c: reallocFGets, myIsReal, strTrim, strToLower
 clock.c: Clock_Epoch2YearDay, Clock_MonthNum,
 Clock_NumDay, Clock_GetTimeZone,
 Clock_IsDaylightSaving, Clock_Seconds,
 Clock_ScanZone, Clock_ScanDate

Return

None

Clock_Epock2YearDay:

Author

Arthur Taylor, September 2002 MDL/RSIS
 AAT, June 2004, (MDL) <Updated>

Purpose

To convert the days since the beginning of the epoch to
 days since beginning of the year and years since the
 beginning of the epoch.

Variables Passed In

totDay : Number of days since the beginning of the
 epoch
 Day : The days since the beginning of the year.
 Yr : The years since the epoch.

Local Variables

None

Subroutine Calls (File:Subroutines)

None

Return

None

Clock_MonthNum:

Author

Arthur Taylor, September 2002 MDL/RSIS
AAT, June 2004, (MDL) <Updated>

Purpose

Determine which numbered month given the day beginning of the year and the year since the beginning of each epoch.

Variables Passed In

Day : Day since the beginning of the year
year : Year since the beginning of the epoch

Local Variables

None

Subroutine Calls (File:Subroutines)

None

Return

int : Determined month

Clock_NumDay:

Author

Arthur Taylor, September 2002 MDL/RSIS

Purpose

Returns either the number of days in the month or the number of days since the beginning of the year

Variables Passed In

month : Month in question
day : Day of month in question
year : Years since the epoch
f_tot : 1 for total days from beginning of year
2 for total days in the month

Local Variables

None

Subroutine Calls (File:Subroutines)

None

Return

int : Either the number of days in the month, or
the number of days since the beginning of
the year.

Clock_FormatParse:

Author

Arthur Taylor, September 2002, MDL/RSIS
ATT, June 2004, MDL <updated>

Purpose

To format part of the output clock string

Variables Passed In

buffer : The output string to write to
sec : Seconds since the beginning of day
floatSec : Part of a second since beginning of second
totDay : Days since the beginning of the epoch
year : Years since the beginning of the epoch
month : Days since the beginning of the year
day : Days since the beginning of the year
format : Which part of the format string being
worked on

Local Variables

MonthName : Array of month names
DayName : Array of day names
dy : Number of days from start of year to start
of month
temp : Array to help parse the %D, %T, %r, and
%R options

Subroutine Calls (File:Subroutines)

clock.c: Clock_NumDay, Clock_FormatParse

Return

None

Clock_GetTimeZone:

Author

Arthur Taylor, June 2004, MDL

Purpose

Returns the time zone offset in hours to adjust to UTC

Variables Passed In

None

Local Variables

buff : Character string holding time
 tm time : Standard C time function struct variable
 ansTime : time_t variable
 timeZone : Number of hours to adjust the clock due to the time zone

Subroutine Calls (File:Subroutines)

None

Return

int : Determined time zone

Clock_IsDaylightSaving:

Author

Arthur Taylor, September 2002, MDL/RSIS
 ATT, June 2004, MDL <updated>

Purpose

Determine if daylight savings is in effect. Daylight savings is in effect from the first Sunday in April to the last Sunday in October.

At 2 AM ST in April -> 3 AM DT (return 1)

At 2 AM DT in October -> 1 AM ST (return 0)

Variables Passed In

clock : The time stored as a double
 TimeZone : Number of hours to adjust the clock due to the time zone

Local Variables

totDay : Days since the beginning of the epoch
 year : Years since the beginning of the epoch
 sec : Seconds since the beginning of day
 day : Days since the beginning of the year
 first : Determines day (mon, tue, wed, ...) of first of year Jan 1

Subroutine Calls (File:Subroutines)

clock.c: Clock_Epoch2YearDay

Return

int : 0 not daylight savings time
 : 1 if is daylight savings time

Clock_PrintDate:

Author

Unknown

Purpose

Determines printable date

Variables Passed In

clock : The time stored as a double
 month : Months since the beginning of the year
 hour : Days since the beginning of the epoch
 min : Minutes since the beginning of the epoch
 year : Years since the beginning of the epoch
 sec : Seconds since the beginning of day
 day : Days since the beginning of the year

Local Variables

totDay : Days since the beginning of the epoch
 intsec : Seconds since the beginning of day

Subroutine Calls (File:Subroutines)

None

Subroutine Calls (File:Subroutines)

clock.c: Clock_Epoch2YearDay, Clock_MonthNum,
 Clock_NumDay

Return

None

Clock_Print:**Author**

Arthur Taylor, September 2002, MDL/RSIS
 ATT, June 2004, MDL <updated>

Purpose

To create formatted output from a time structure that is stored as a double

Variables Passed In

buffer : Destination to write the format to
 n : The number of characters in the buffer
 (Input)
 clock : The time stored as a double
 format : The desired output format
 f_gmt : 0 output GMT, 1 output LDT, 2 output
 LST (Input)

Local Variables

totDay : Days since the beginning of the epoch
 year : Years since the beginning of the epoch

sec : Seconds since the beginning of day
 floatSec : Seconds since the beginning of the day
 month : Numeric month
 day : Numeric day
 i, j : For loop counting numbers
 f_perc : Case statement switch bit
 locBuff : Character array containing time
 timeZone : Number of hours to adjust the clock by due
 to time zone
 clock.c: Clock_Epoch2YearDay, Clock_MonthNum,
 Clock_FormatParse, Clock_GetTimeZone,
 Clock_IsDaylightSavings

Return

None

Clock_Clicks:

Author

Arthur Taylor, June 2002, MDL/RSIS

Purpose

Returns the number of time/clock click since the program started running.

Variables Passed In

None

Local Variables

ans : Determined clock clicks

Subroutine Calls (File:Subroutines)

None

Return

double : number of clock clicks in var ans

Clock_Seconds:

Author

Arthur Taylor, June 2002, MDL/RSIS

Purpose

To returns the current number of seconds since the beginning of the epoch. Using the local system time zone

Variables Passed In

None

Local Variables

ans : Determined number of seconds

Return

None

Clock_ScanZone:

Author

Arthur Taylor, September 2002, MDL/RSIS
ATT, June 2004, MDL <updated>

Purpose

Scans a character string to determine the time zone

Variables Passed In

ptr : The character string to scan
TimeZone : Number of hours to shift from UTC
f_day : True if we are dealing with daylight
savings

Local Variables

ans : Determined clock clicks

Subroutine Calls (File:Subroutines)

None

Return

int : 0 if time zone found, -1 if not

Clock_ScanMonth:

Author

Arthur Taylor, September 2002, MDL/RSIS
ATT, June 2004, MDL <updated>

Purpose

Scans a string for a month word and assumes string is all caps

Variables Passed In

ptr : The character string to scan

Local Variables

None

Subroutine Calls (File:Subroutines)

None

Return

int : Returns the month number read, or -1 if no month would found

Clock_ScanWeekday:

Author

Arthur Taylor, September 2002, MDL/RSIS
ATT, June 2004, MDL <updated>

Purpose

Scans a string looking for a day word and assumes string is in all caps

Variables Passed In

ptr : The character string to scan

Local Variables

None

Subroutine Calls (File:Subroutines)

None

Return

int : Returns the month number read, or -1 if no month would found

Clock_ScanColon:

Author

Arthur Taylor, September 2002, MDL/RSIS
ATT, June 2004, MDL <updated>

Purpose

Parses a word assuming it is “.” separated and is dealing with hours:minutes:seconds or hours:minutes. Returns the resulting time as a double

Variables Passed In

ptr : The character string to scan

Local Variables

hour : Days since the beginning of the epoch
min : Minutes since the beginning of the epoch
sec : Retrieved time in seconds
ptr3 : Temp variable

Subroutine Calls (File:Subroutines)

None

Return

double : The time after converting the “.” delineated string

Clock_ScanSlash:

Author

Arthur Taylor, September 2002, MDL/RSIS
ATT, June 2004, MDL <updated>

Purpose

Parses a word assuming it is delineated by a forward slash “/” and is in a string format months/days/years or months/days

Variables Passed In

word : The character string to scan
mon : The month that was seen
day : Numeric day of month
year : Numeric year
f_year : true or 1 if year is valid, 0 if not

Local Variables

ptr3 : Temp string pointer
ptr : Pointer to “word” variable defined above

Subroutine Calls (File:Subroutines)

None

Return

int : -1 if month or day is out of range
0 if no problems or success

Clock_ScanDate:

Author

Unknown

Purpose

Scan for date in “clock” variable

Variables Passed In

clock : The time stored as a double
year : Years since the beginning of the epoch
mon : Numeric day of month
day : Numeric year

Local Variables

i : Undefined variable
delt : Difference between 1970 (temp variable)
and current year

temp : Year 1970 start of clock time
totDay : Days since the beginning of the epoch

Subroutine Calls (File:Subroutines)

myassert.c: myAssert
clock.c: Clock_NumDay

Return

None : However “clock” variable pointer passed in
is altered (global) through routine

Clock_GetWord:

Author

Unknown

Purpose

Getting 30 character value for “word” variable. Exact process: 1) Start at \$start, advance start until it is at first non-space, non- ‘,’ non-‘,’ character. Move to end of first space, ‘,’ or ‘.’ After new start location. Copy up to 30 characters (in caps) into word.

Variables Passed In

start : Pointer to start of string
end : Pointer to end of string
word : 30 character array of determined
word/value
wordtype : 0 = none
1 = “.” word
2 = “/”
3 = integer word
4 = AM/PM word
5 = timeZone word
6 = month word
7 = weekday word
8 = Preceding to a relativeDate word
9 = Post to a relativeDate word
10 = relativeDate word unit
11 = Adjust Day word

Local Variables

ptr : TBD
cnt : TBD
f_integer : TBD

Subroutine Calls (File:Subroutines)

None

Return

int : -1 if no next word
 0 if there is or success

Clock Scan:**Author**

Unknown

Purpose

TBD

Variables Passed In

clock : Pointer to the time stored as a double
 buffer : TBD
 f_gmt : TBD

Local Variables

ptr : TBD
 ptr2 : TBD
 word : 30 character array of determined word/value
 wordtype : 0 = none
 1 = “.” word
 2 = “/”
 3 = integer word
 4 = AM/PM word
 5 = timeZone word
 6 = month word
 7 = weekday word
 8 = Preceding to a relativeDate word
 9 = Post to a relativeDate word
 10 = relativeDate word unit
 11 = Adjust Day word
 TimeZone : TBD
 lastwordType : TBD
 f_dayLight : TBD
 month : TBD
 day : TBD
 year : TBD
 f_year : TBD
 index : TBD
 ans : TBD
 Stack : struct of StackType
 Rel : struct of relType
 lenRel : TBD
 lenStack : TBD

PreRel : Static char array containing LAST, THIS, NEXT, NULL words used during product generation printouts
 PreUnit : Static char array containing YEAR, YEARS, MONTH, MONTHS, FORTNIGHT, FORTNIGHTS, WEEK, WEEKS, DAY, DAYS, HOUR, HOURS, MIN, MINS, MINUTE, MINUTES, SEC, SEC, SECOND, SECONDS, NULL used during product generation printouts.
 AdjDay : Static char array containing YESTERDAY, TODAY, TOMORROW, NULL used as adjusted day for product generation printouts
 f_ampm : TBD
 f_timeZone : TBD
 f_time : TBD
 f_date : TBD
 f_slashWord : TBD
 f_dateWord : TBD
 f_monthWord : TBD
 f_dayword : TBD
 curTime : Variable containing current time
 sec : TBD
 i : TBD
 monthAdj : TBD
 yearAdj : TBD

Subroutine Calls (File:Subroutines)

myUtil.c: GetIndexFromStr,
 clock.c: Clock_Epoch2YearDay, Clock_MonthNum, Clock_NumDay, Clock_GetTimeZone, Clock_Seconds, Clock_ScanZone, Clock_ScanMonth, Clock_ScanWeekday, Clock_ScanColon, Clock_ScnSlash, Clock_ScanDate, Clock_GetWord, Clock_IsDaylightSaving

Return

None

3.4.2

Source file “complex.c / complex.h” Subroutine

Comp_sqrt:

Author

Unknown

Purpose

Calculate square root of given number

Variables Passed In

z : complex_type struct of x & y double values defined in “Global Variables” section

Local Variables

ans : Calculated answer of complex_type struct
r : Summation of x & y squares

Subroutine Calls (File:Subroutines)

None

Return

Complex_type: **TBD**

Comp_log:

Author

Unknown

Purpose

TBD

Variables Passed In

z : complex_type struct of x & y double values defined in “Global Variables” section

Local Variables

ans : Calculated answer of complex_type struct

Subroutine Calls (File:Subroutines)

None

Return

Complex_type: **TBD**

Comp_exp:

Author

Unknown

Purpose

TBD

Variables Passed In

z : complex_type struct of x & y double values defined in “Global Variables” section

Local Variables

ans : Calculated answer of complex_type struct

Subroutine Calls (File:Subroutines)

None

Return

Complex_type: TBD

Comp_mul:

Author

Unknown

Purpose

TBD

Variables Passed In

z : complex_type struct of x & y double values defined in “Global Variables” section

Local Variables

ans : Calculated answer of complex_type struct

Subroutine Calls (File:Subroutines)

None

Return

Complex_type: TBD

Comp_R_mul:

Author

Unknown

Purpose

TBD

Variables Passed In

z1 : complex_type struct of x & y double
values defined in “Global Variables”
section
z2 : complex_type struct of x & y double
values defined in “Global Variables”
section

Local Variables

ans : Calculated answer of complex_type
struct

Subroutine Calls (File:Subroutines)

None

Return

Complex_type: TBD

Comp_inv:

Author

Unknown

Purpose

TBD

Variables Passed In

z : complex_type struct of x & y double
values defined in “Global Variables”
section

Local Variables

ans : Calculated answer of complex_type
struct

Subroutine Calls (File:Subroutines)

None

Return

Complex_type: TBD

Comp_add:

Author

Unknown

Purpose

Add like variables x & y provided in 2 complex type structs

Variables Passed In

z1 : complex_type struct of x & y double
values defined in “Global Variables”
section

z2 : complex_type struct of x & y double
values defined in “Global Variables”
section

Local Variables

ans : Calculated answer of complex_type
struct

Subroutine Calls (File:Subroutines)

None

Return

Complex_type: **TBD**

Comp_set:

Author

Unknown

Purpose

Subtract to like variables x & y provided in 2 complex
type structs

Variables Passed In

x : x coordinate

y : y coordinate

Local Variables

ans : Calculated answer of complex_type
struct

Subroutine Calls (File:Subroutines)

None

Return

Complex_type: **TBD**

Comp_real:

Author

Unknown

Purpose

Return x variable value in a provided complex_type struct

Variables Passed In

z : complex_type struct of x & y double values defined in “Global Variables” section

Local Variables

None

Subroutine Calls (File:Subroutines)

None

Return

Complex_type : Value of x in provided complex_type structure

Comp_imag:

Author

Unknown

Purpose

Return y variable value in a provided complex_type struct

Variables Passed In

z : complex_type struct of x & y double values defined in “Global Variables” section

Local Variables

None

Subroutine Calls (File:Subroutines)

None

Return

Complex_type : Value of y in provided complex_type structure

Comp_print:

Author

Unknown

Purpose

Print to standard out x & y variable values in provided complex_type structure.

Variables Passed In

z : complex_type struct of x & y double values defined in “Global Variables” section

Local Variables

None

Subroutine Calls (File:Subroutines)

None

Return

None

3.4.3**Source file “cstart.c” Subroutine**main:**Author**

Unknown

Purpose

TBD

Variables Passed In

Argc : TBD

Argv : TBD

Local Variables

fp : TBD

buffer : TBD

lenPtr : TBD

cnt : TBD

i : TBD

lineArgc : TBD

lineArgv : TBD

bsnAbrev : TBD

f_DoIt : TBD

imxb : TBD

jmx b : TBD

grid : TBD

rexName : TBD

envName : TBD

Subroutine Calls (File:Subroutines)

NA

Return

TBD

CopyBasin:**Author**

Unknown

Purpose**TBD****Variables Passed In**

bsnAbrev : **TBD**
 bsnDataDir : **TBD**
 dtaName : **TBD**
 llxName : **TBD**
 imxb : **TBD**
 jbxb : **TBD**

Local Variables

len : **TBD**
 type : **TBD**
 nameLen : **TBD**
 fileName : **TBD**
 fp : **TBD**
 buffer : **TBD**
 lenPtr : **TBD**
 bsnRoot : **TBD**
 lineRoot : **TBD**
 f_found : **TBD**

Subroutine Calls (File:Subroutines)

myUtil.c: reallocFGets, FileCopy, strToLower
 savellx.c: saveLLX
 myassert.c: myAssert

Return**TBD**setUpFiles:**Author**

Unknown

Purpose**TBD****Variables Passed In**

bsn : **TBD**
 track : **TBD**
 rex : **TBD**
 bsnAbrev : **TBD**
 slshBsnDir : **TBD**
 trkName : **TBD**
 dtaName : **TBD**

llxName : TBD
 rexName : TBD
 imxb : TBD
 jmx b : TBD

Local Variables

None

Subroutine Calls (File:Subroutines)

myUtil.c: FileCopy, CopyBasin
 myassert.c: myAssert

Return

TBD

PreformRun:

Author

Unknown

Purpose

TBD

Variables Passed In

bsnAbrev : TBD
 dtaName : TBD
 llxName : TBD
 trkName : TBD
 xxxName : TBD
 rexName : TBD
 imxb : TBD
 jmx b : TBD

Local Variables

st : TBD
 mhalt : End interval
 clock : TBD
 etime : TBD
 rextime : TBD
 f40Name : TBD
 f_RexReset : TBD
 f_env : TBD
 itime : Interval time
 f_first : TBD
 rexFp : TBD
 del_t : TBD
 csflag : Flag to do CS computations
 f_graphics : Copy data/stuff, but don't copy graphics

f_passdata : TBD
f_smooth : TBD
f_saveEnv : TBD

Subroutine Calls (File:Subroutines)

Slosh2.c: SaveRexStep, RunLoopStep, RunInit

Return

TBD

DoOneStorm:

Author

Unknown

Purpose

TBD

Variables Passed In

bsn : TBD
slshBsnDir : TBD
track : TBD
rex : TBD

Local Variables

i : TBD
imxb : TBD
jmxnb : TBD
grid : TBD
rexName : TBD
bsnAbrev : TBD

Subroutine Calls (File:Subroutines)

myUtil.c: FileCopy
csstart.c: PerformRun

Return

TBD

3.4.4

Source file “myassert.c / myassert.h” Subroutine

myAssert:

Author

Arthur Taylor, Agust 2003, MDL/RSIS

Purpose

This is an assert routine from “Writing Solid Code” by Steve Maguire.

Advantages of this over C's "assert" function is that assert stores the expression string for printing. Storing is believe to be done in a global data value, but that means assert is using memory space that the SLOSH program may need for read advantages. If you trigger assert, you're ok to integrate the file and see the code.

Variables Passed In

file : Filename that assert was in
lineNum : Line number in file of assert

Local Variables

None

Subroutine Calls (File:Subroutines)

None

Return

None

3.4.5 Source file "myutil.c / myutil.h" Subroutine

Global Variables: POWERS_ONE : Array of "powers of" values from 1e0 to 1e17

reallocFGets:

Author

Arthur Taylor, December 2002, MDL/RSIS

Purpose

Read file a passed in until end of line \n char reached. Reallocate memory as needed. Similar to fgets, except we don't know ahead of time that the line is a specific length. Assumes that Ptr is either NULL, or points to lenBuff memory. Responsibility of caller to free the memory

Notes:

- 1) Based on getline (see K&R C book (2nd edition) p 29) and on the behavior of TCL's get routine
- 2) Chose MIN_STEPSIZE = 80 because pages are usually 90 columns
- 3) could switch lenBuff = I + 1 / lenBuff = I to always true. Rather not...less allocs... This way code behaves almost the same as fgets except it can expand as needed.

Variables Passed In

Ptr : An array of data that is of size LenBuff
 LenBuff : The allocated length of Ptr
 fp : Input file stream, file pointer.

Local Variables

buffer : Local copy of Ptr as passed into routine
 lenBuff : Local copy of LenBuff as passed into routine
 c : Current char read in from the data stream (file)
 i : Array increment, where to store character written to variable “c”

Subroutine Calls (File:Subroutines)

None

Return

int : strlen (buffer)...
 0 for read only EOF
 1 for “\nEOF” or “<char>EOF”

mySplit:

Author

Arthur Taylor, May 2004, MDL/RSIS

Purpose

Split a character array according to a given symbol.
 Responsibility of caller to free allocated memory.

Variables Passed In

Data : Character string to process
 Symbol : Character to split string at
 Argc : Number of groupings found
 Argv : Characters in each grouping
 f_trim : Set to true (1) if white space exists in string and needing to be trimmed

Local Variables

head : Pointer to beginning of integrated string
 ptr : Used to mark the position of any existing symbols passed by the user in variable “Symbol”
 argc : Local copy of Argc
 argv : Local copy of Argv

Subroutine Calls (File:Subroutines)

myUtil.c: strTrim

Return

None

MyIsReal:**Author**

Arthur Taylor, July 2004, MDL/RSIS

Purpose

Returns 0 if all characters are digits except a trailing “,”, or leading character of “-“. Value is set to atof (ptr)

Variables Passed In

ptr : Character string to integrate
 value : The converted value of ptr, if ptr is a number

Local Variables

len : Length of string pointer ptr
 i : for loop counter

Subroutine Calls (File:Subroutines)

None

Return

int : 0 if it is not a real number
 1 if it is a real number

FileCopy:**Author**

Arthur Taylor, May 2004, MDL/RSIS

Purpose

Copy a file from one location to another

Variables Passed In

fileIn : Source file to read from
 fileOut : Destination file

Local Variables

ifp : FILE in file pointer
 ofp : FILE output pointer
 c : Temp character holder from reading to writing

Subroutine Calls (File:Subroutines)

None

Return

int : -1 if error
0 for success

FileTail:

Author

Arthur Taylor, May 2004, MDL/RSIS

Purpose

Returns the characters in a filename after the last directory separator. Responsibility of call to free memory.

Variables Passed In

fileName : Filename to read from
tail : end of filename

Local Variables

ptr : Navigation pointer, moving through string
based on static symbols “\\” or “\”

Subroutine Calls (File:Subroutines)

None

Return

None

MyRound:

Author

Arthur Taylor, May 2003, MDL/RSIS

Purpose

Round a number to a given number of decimal places

Variables Passed In

data : Number to round
place : Rounding decimal place

Local Variables

None

Subroutine Calls (File:Subroutines)

None

Return

double : Rounded value passed into routine

strTrim:

Author

Arthur Taylor, October 2003, MDL/RSIS

Purpose

To trim white spaces from both sides of a character string

Variables Passed In

str : Character string pointer to integrate

Local Variables

i : Loop counter to traverse string

j : Loop counter to traverse string

len : String length

Subroutine Calls (File:Subroutines)

None

Return

None

strToUpper:**Author**

Arthur Taylor, October 2003, MDL/RSIS

Purpose

To convert a string to all uppercase characters

Variables Passed In

str : Character string pointer to integrate

Local Variables

ptr : Pointer containing converted character
string

Subroutine Calls (File:Subroutines)

Myassert.c: myAssert

Return

None

GetIndexFromStr:**Author**

Arthur Taylor, October 2003, MDL/RSIS

Purpose

Given a search word/key parse provided string and return
index of word/key location

Variables Passed In

arg : The string to integrate, parsed string

Opt : The word/key to find in string

Index : The location of arg in Opt (or -1 if not located)

Local Variables

ptr : local pointer navigation pointer initialized to Opt variable
cnt : Beginning location of word/key found

Subroutine Calls (File:Subroutines)

None

Return

None

strToLower:

Author

Arthur Taylor, May 2004, MDL/RSIS

Purpose

To convert a string to all lower case characters

Variables Passed In

Str : The string to adjust

Local Variables

ptr : local pointer containing result of converted string in lower case characters

Subroutine Calls (File:Subroutines)

myassert.c: myAssert

Return

None

3.4.6

Source file “pack.c / pack.h” Subroutine

Global Variables

ltln_ray : **TBD**
Ltln_val : **TBD**

main:

Author

Arthur Taylor, April 2003, MDL/RSIS

Purpose

To test the memBitRead and memBitWrite routines, to make sure that they function correctly on some sample data

Variables Passed In

argc : Number of command line arguments
argv : Command line arguments – non passed

Local Variables

buff : TBD
buff2 : TBD
bufLoc : TBD
ptr, ptr2 : TBD
numUsed : TBD

Subroutine Calls (File:Subroutines)

NA

Return

int : 1 for error, 0 for success

Stuff_XXX:

Author

Arthur Taylor, April, 20 1998, MDL/RSIS

Purpose

To stuff/fill basin data, using current scheme for basin.

Note:

Uses a pcx like encoding scheme. Huffman codes are only good if we have a delta? A better method might use delta's for non-land flags. Also use huff codes for run lengths. Simple scheme... Always print 8 bit fixed run [1..256] followed by fixed 9 bit literal –15.0..36.0] literal 999 = 36.1. for speed sake may want entire array passed in.

Variables Passed In

fp : File pointer of opened binary file, using TIO.c
pbuf : Character pointer buffer to hold file contents when reading
pubfLoc : Location of the bit buffer
val : The slosh value to short (10*height or 999)
f_flag : 0 => normal 1-> flush the stream and bit_buffer

Local Variables

prev : Data to write to file
run : Data to write to file
count : Stores number of bits saved or -1 if error

Subroutine Calls (File:Subroutines)

tedian.c: fileBitWrite

Return

int : Value of count, number of bits saved or -1 if error

memStuff_xxx:

Author

unknown

Purpose

Determine which bits of a data structure to write, actual writing is done in memBitWrite

Variables Passed In

ptr : Character buffer to write to
bufLoc : How many bites to write
val : The slosh value to short (10*height or 999)
f_flag : 0 => normal 1-> flush the stream and
bit_buffer

Local Variables

prev : Data to write to file
run : Data to write to file
count : Stores number of bits saved or -1 if error
numUsed : How many bytes were written to Dst
variable (output char buffer)

Subroutine Calls (File:Subroutines)

tedian.c: memBitWrite

Return

int : Value of count, number of bits saved or -1 if error

Stuff2_xxx:

Author

Arthur Taylor, April, 20 1998, MDL/RSIS

Purpose

To stuff/fill basin data, using current scheme for basin.

Note:

Uses a pcx like encoding scheme. Huffman codes are only good if we have a delta? A better method might use delta's for non-land flags. Also use huff codes for run lengths. Simple scheme... Always print 8 bit fixed run [1..256]

followed by fixed 10 bit literal [-32.0..70.0] literal 999 = 70.1. for speed sake may want entire array passed in.

Variables Passed In

fp : File pointer of opened binary file, using TIO.c
pbuf : Character pointer buffer to hold file contents when reading
pubfLoc : Location of the bit buffer
val : The slosh value to short (10*height or 999)
f_flag : 0 => normal 1-> flush the stream and bit_buffer

Local Variables

prev : Data to write to file
run : Data to write to file
count : Stores number of bits saved or -1 if error

Subroutine Calls (File:Subroutines)

tendian.c: fileBitWrite

Return

int : Value of count, number of bits saved or -1 if error

memStuff2 xxx:

Author

unknown

Purpose

Determine which bits of a data structure to write, actual writing is done in memBitWrite

Variables Passed In

ptr : Character buffer to write to
bufLoc : How many bites to write
val : The slosh value to short (10*height or 999)
f_flag : 0 => normal 1-> flush the stream and bit_buffer

Local Variables

prev : Data to write to file
run : Data to write to file
count : Stores number of bits saved or -1 if error
numUsed : How many bytes were written to Dst variable (output char buffer)

Subroutine Calls (File:Subroutines)

tendian.c: memBitWrite

Return

int : Value of count, number of bits saved or -1 if error

UnStuff_xxx:

Author

Arthur Taylor, April, 20 1998, MDL/RSIS

Purpose

To unstuff the basin with data, remove data from initialized basin

Note:

Uses a pcx like encoding scheme. Huffman codes are only good if we have a delta? A better method might use delta's for non-land flags. Also use huff codes for run lengths. Simple scheme... Always print 8 bit fixed run [1..256] followed by fixed 9 bit literal -15.0..36.0] literal 999 = 36.1. for speed sake may want entire array passed in.

Variables Passed In

fp : File pointer of opened binary file, using TIO.c
gbuf : Character pointer buffer to hold file contents when reading
gbufLoc : Location of the bit buffer
val : The slosh value to short (10*height or 999)
f_flag : 0 => normal 1-> flush the stream and bit_buffer

Local Variables

prev : Data read from buffer
run : Data read from buffer
temp : Temp data read from buffer

Subroutine Calls (File:Subroutines)

tendian.c: fileBitRead

Return

int : 1 success, 0 fail

UnStuff2_xxx:

Author

Arthur Taylor, April, 20 1998, MDL/RSIS

Purpose

To unstuffy the basin with data, remove data from initialized basin

Note:

Uses a pcx like encoding scheme. Huffman codes are only good if we have a delta? A better method might use delta's for non-land flags. Also use huff codes for run lengths. Simple scheme... Always print 8 bit fixed run [1..256] followed by fixed 10 bit literal [-32.0..70.0] literal 999 = 70.1. for speed sake may want entire array passed in.

Variables Passed In

fp : File pointer of opened binary file, using TIO.c
 gbuf : Character pointer buffer to hold file contents when reading
 gbufLoc : Location of the bit buffer
 val : The slosh value to short (10*height or 999)
 f_flag : 0 => normal 1-> flush the stream and bit_buffer

Local Variables

prev : Data read from buffer
 run : Data read from buffer
 temp : Temp data read from buffer

Subroutine Calls (File:Subroutines)

tendian.c: fileBitRead

Return

int : 1 success, 0 fail

UnStuff_LatLon:**Author**

Arthur Taylor, April, 20 1998, MDL/RSIS

Purpose

This routine uses the Huffman code chosen for lat/lon files, and methods such as [10000] for exchange in sign, [0][5bit+extra] for literal [1][4bits+extra][4bits+extra] for length, distance pair

Variables Passed In

fp : File pointer of opened binary file, using TIO.c

gbuf : Character pointer buffer to hold file
 contents when reading
 gbufLoc : Location of the bit buffer
 x : TBD
 sgn : TBD
 f_flag : 0 normal, 1 flush, 2 stream

Local Variables

lit : The storage place for the data read from
 file
 index : TBD
 lit2 : The storage place for the data read from
 file
 first : TBD
 second : TBD
 enc_x : TBD
 last : TBD
 len : -1 if sistance is leteral

Subroutine Calls (File:Subroutines)

tendian.c: fileBitRead

Return

int : 0 for success, -1 fail

3.4.7

Source file “savellx.c / savellx.h” Subroutine

cl2cnf:

Author

Arthur Taylor, January, 29 1998, MDL/RSIS

Purpose

Convert a latitude on the Clarke sphere (regular earth latitude) to one on the conformal sphere (a perfect sphere from the clarke sphere by pulling on the poles)

Variables Passed In

clk : The Clarke latitude
 cnf : The conformal latitude
 scale : The local scale of the map between
 spheroids

Local Variables

sin_clk : TBD
 temp0 : TBD
 temp1 : TBD

temp2 : TBD

Subroutine Calls (File:Subroutines)

None

Return

None

cnf2cl:

Author

Arthur Taylor, January, 29 1998, MDL/RSIS

Purpose

Convert a latitude on the conformal sphere (a perfect sphere from the Clarke sphere by pulling on the poles) to the clarke sphere (regular earth latitude. Also returns a local scale of the map between spheroids. The scale should be a ration of distance on conformal sphere to clarke sphere and should be ≥ 1.0 .

Variables Passed In

clk : The Clarke latitude
cnf : The conformal latitude
scale : The local scale of the map between
spheroids

Local Variables

sin_clk : TBD
temp0 : TBD
temp1 : TBD
temp2 : TBD
temp_lat : TBD
grad : TBD
i : TBD

Subroutine Calls (File:Subroutines)

None

Return

None

Sll2xy:

Author

Arthur Taylor, January, 29 1998, MDL/RSIS

Purpose

Convert a latitude on the conformal sphere (a perfect sphere from the Clarke sphere by pulling on the poles) to

the clarke sphere (regular earth latitude. Also returns a local scale of the map between spheroids. The scale should be a ration of distance on conformal sphere to clarke sphere and should be ≥ 1.0 .

Variables Passed In

clk : The Clarke latitude
cnf : The conformal latitude
scale : The local scale of the map between
spheroids

Local Variables

sin_clk : TBD
temp0 : TBD
temp1 : TBD
temp2 : TBD
temp_lat : TBD
grad : TBD
i : TBD

Subroutine Calls (File:Subroutines)

None

Return

None

pq2xy:

Author

Arthur Taylor, January, 29 1998, MDL/RSIS

Purpose

Converts p, q coordinates to x, y coordinates on the SLOSH tangent plane

Variables Passed In

p, q : The p, q coordinates in the SLOSH grid
x, y : The x, y coordinates on the SLOSH
tangent plane
bsn : The defining parameters of the basin

Local Variables

Z1 : TBD
Z2 : TBD
xa : TBD
ya : TBD
x1 : TBD

y1 : TBD
cs : TBD
ss : TBD

Subroutine Calls (File:Subroutines)

complex.c: Comp_exp, Comp_R_Mul, Comp_inv,
Comp_add, Comp_Set, Comp_real, Comp_imag

Return

None

xy2pq:

Author

Arthur Taylor, January, 30 1998, MDL/RSIS

Purpose

Converts x, y coordinates on the SLOSH tangent plane to
SLOSH p, q coordinates

Variables Passed In

p, q : The p, q coordinates in the SLOSH grid
x, y : The x, y coordinates on the SLOSH
tangent plane
bsn : The defining parameters of the basin

Local Variables

Z1 : TBD
Z2 : TBD
xa : TBD
ya : TBD
x1 : TBD
y1 : TBD
cs : TBD
ss : TBD

Subroutine Calls (File:Subroutines)

clock.c: Comp_sqrt, comp_log, Comp_mul, Comp_R_mul,
Comp_add, Comp_set, Comp_real, Comp_imag

Return

None

Fort_atof:

Author

Arthur Taylor, January, 29 1998, MDL/RSIS

Purpose

Converts a string to a double using FORTRAN conventions

Note:

1) Apparently FORTRAN does not put zeros in the blanks if it is reading an integer into a F4.1 statement. Also, F4.5 is allowed. This was checked with both Lahey and Microsoft DOS FORTRAN compilers

Variables Passed In

s : The string to extract the double from
w : The FORTRAN field width
d : The FORTRAN decimal field

Local Variables

c : TBD
ans : TBD

Subroutine Calls (File:Subroutines)

None

Return

double : Returns variable “s” using FORTRAN conventions

a2lat:

Author

Arthur Taylor, January 1998, MDL/RSIS
Arthur Taylor, January 1999, MDL <updated>
Arthur Taylor, May 2004, MDL <updated>

Purpose

Converts a string containing deg min sec for latitude to a double

Variables Passed In

s : The string to get latitude from

Local Variables

deg : TBD
min : TBD
sec : TBD
ans : TBD
buffer : TBD
c : TBD
tmp : TBD

Subroutine Calls (File:Subroutines)

None

Return

double : Determined latitude string

str2bsndta:**Author**

Arthur Taylor, January 1998, MDL/RSIS
 Arthur Taylor, May 2004, MDL <updated>

Purpose

Converts a NULL terminated string containing a line from asins.dta file into the bsndta type. Also, converts this read at/long from Clarke spheroid to conformal sphere, and alizes the tangent plane

Variables Passed In

s : The NULL terminated string (assumed lowercase)
 bsn : The resulting bsndta type
 slsh : Holds the math for the SLOSH tangent plane

Local Variables

temp : **TBD**

Subroutine Calls (File:Subroutines)

savellx.c: cl2cnf, sl2xy, Fort_atof, a2lat

Return

None

saveLLx:**Author**

Arthur Taylor, January 1998, MDL/RSIS

Purpose

TBD

Variables Passed In

buffer : Contains the line from ('', e/h) basins.dta
 filename : is where to save the llx file data to
 Header : **TBD**
 Imxb : **TBD**
 Jmxb : **TBD**

Local Variables

bsm : **TBD**
 slsh : **TBD**

temp1	: TBD
temp2	: TBD
fp	: TBD
header	: TBD
temp	: TBD
scale	: TBD
x1	: TBD
y2	: TBD
inxb	: TBD
jmx	: TBD
DX	: TBD
i	: TBD
j	: TBD

Subroutine Calls (File:Subroutines)

savellx.c: cnf2cl, sxy2ll, pq2xy, str2bsnda

Return

int : -1 fail, 0 success

3.4.8

Source file “slosh2.c / slosh2.h” Subroutine

Data_WriteTrk:

Author

Unknown

Purpose

TBD

Variables Passed In

Fp2	: TBD
trk_name	: File name string/pointer

Local Variables

buff1	: TBD
buff2	: TBD
fp	: File pointer for trk_name file name
lat	: TBD
long	: TBD
spd	: TBD
dir	: TBD
delp	: TBD
rmax	: radius of maximum (storm) winds
I	: TBD
j	: TBD
k	: TBD
c_temp	: TBD

f_temp : TBD

Subroutine Calls (File:Subroutines)

tendian.c: revfwrite

Return

TBD

SaveRexHeader:

Author

Unknown

Purpose

TBD

Variables Passed In

fp : TBD

imxb : TBD

jmx b : TBD

f_type : TBD

offset : TBD

track_name : TBD

abbrev : TBD

min : TBD

max : TBD

Local Variables

fp2 : TBD

i

str_len : TBD

name_len : TBD

si_temp : TBD

sj_temp : TBD

s_temp : TBD

c_temp : TBD

temp : TBD

l_temp : TBD

Subroutine Calls (File:Subroutines)

tendian.c: revfwrite

Return

TBD

SaveRexStep:

Author

Unknown

Purpose

TBD

Variables Passed In

f_restoffset : **TBD**
fp : **TBD**
gt : **TBD**
imxb : **TBD**
jmx b : **TBD**
grid : **TBD**
f_type : **TBD**
trkName : **TBD**
bsnAbrev : **TBD**
f_env : **TBD**
clock : **TBD**

Local Variables

offset : **TBD**
trkoffset
i : **TBD**
j : **TBD**
val : **TBD**
bits : **TBD**
l_temp : **TBD**
min_h : **TBD**
max_h : **TBD**
year : **TBD**
mon : Numeric (int) of the month
day : Numeric (int) day of the month
hour : Numeric (int **24 clock TBD?**) hour
min : Numeric int minutes
sec : Numeric double seconds (full time)
si_temp : **TBD**
c_temp : **TBD**
f_temp : **TBD**

Subroutine Calls (File:Subroutines)

myUtil.c: myRound,
tendian.c: Stuff_XXX, Sutff2_XXX
slosh2.c: Data_WriteTrk, SaveRexHeader
clock.c: Clock_PrintDate

Return

TBD

RunLoopStep:

Author

Unknown

Purpose

TBD

Variables Passed In

gt	: TBD
imxb	: TBD
jmx	b : TBD
grid	: TBD
itime	: TBD
mhalt	: TBD
csflag	: TBD
f_smooth	: TBD
f_graphics	: TBD
f_passdata	: TBD
del_t	: TBD

Local Variables

i	: TBD
j	: TBD

Subroutine Calls (File:Subroutines)

interface.f: TMSTEP

Return

TBD

CleanUp:

Author

Unknown

Purpose

TBD

Variables Passed In

gt	: TBD
imxb	: TBD
jmx	b : TBD
grid	: TBD
f_saveEnv	: TBD

Local Variables

i	: TBD
j	: TBD

Subroutine Calls (File:Subroutines)

interface.f: CLNUP

Return

TBD

RunInit:

Author

Unknown

Purpose

TBD

Variables Passed In

gt	: TBD
trkName	: TBD
dtaName	: TBD
xxxName	: TBD
llxName	: TBD
ft40Name	: TBD
mhalt	: TBD
clock	: TBD

Local Variables

imxb	: TBD
jmx	: TBD
ylt[][]	: TBD
ylg[][]	: TBD
year	: TBD
mon	: Numeric (int) of the month
day	: Numeric (int) day of the month
hour	: Numeric (int 24 clock TBD?) hour
min	: Numeric int minutes

Subroutine Calls (File:Subroutines)

interface.f: INITAL

Return

TBD

3.4.9

Source file “tendian.c / tendian.h” Subroutine

main:

Author

Arthur Taylor, April 2003, MDL/RSIS

Purpose

To test the memBitRead, and memBitWrite routines making sure they function correctly on sample data.

Variables Passed In

argc : TBD
argv : TBD

Local Variables

buff : TBD
buff2 : TBD
bufLoc : TBD
ptr : TBD
ptr2 : TBD
numUsed : TBD

Subroutine Calls (File:Subroutines)

tendian.c: memBitRead, memBitWrite

Return

int : 1 for error, 0 for success

memswp:**Author**

Arthur Taylor, September 2002, MDL/RSIS

Purpose

To swap memory in the Data array based on the knowledge that there are “num_elem” elements, each of size “elem_size”

Note:

1) A similar routine was provided with the GRIB2 library. It called “unpk_swap”. Since it had the restriction that it only dealt with long ints, this function was created as more flexibility was needed. In addition this procedure may be more efficient. Tests show an operation count for swapping an array that consists of 1 long int. “unpk_swap” = 46 operations, “memswp” = 33.

Variables Passed In

Data : Pointer to the data being swapped
elem_size : The size of an individual element
num_elem : The number of elements to swap

Local Variables

h : Element count
data : Used to treat data as a char array
temp : Temporary holder of a byte when swapping
ptr, ptr2 : Pointers to the two bytes to swap

Subroutine Calls (File:Subroutines)

None

Return

None

revmemcpy:

Author

Arthur Taylor, September 2002, MDL/RSIS

Purpose

To copy memory similar to memcpy, but in a reverse manner. In order to have the same arguments as memcpy, this can not handle arrays... For arrays use revmemcpyRay() function.

Note:

- 1) This function came about as an improvement to improve memcpy. It was assumed this function would run faster than memcpy followed by memswap.
- 2) Assumes that Dst is allocated to a size of Src
- 3) Problems with MEMCPY if len != sizeof(dst)... Is it left or right justified.

Variables Passed In

Dst : The destination for data output
Src : The source of the data
len : The length of Src in bytes

Local Variables

J : Byte count
src : Allows Src to be treated as array src
dst : Allows Dst to be treated an array dst

Subroutine Calls (File:Subroutines)

None

Return

Dst : (Void) Data destination pointer

revmemcpyRay:

Author

Arthur Taylor, September 2002, MDL/RSIS

Purpose

To copy memory similar to memcpy, but in a reverse manner. This handles the case when we need to reverse memcpy an array of data

Note:

1) Assumes that Dst is allocated to a size of Src

Variables Passed In

Dst : The destination for data output
Src : The source of the data
len : The length of Src in bytes

Local Variables

J : Byte count
src : Allows Src to be treated as array src
dst : Allows Dst to be treated as array dst

Subroutine Calls (File:Subroutines)

None

Return

Dst : (Void) Data destination pointer

revfread:

Author

Arthur Taylor, September 2002, MDL/RSIS

Purpose

To do an “fread”, but in a reverse manner

Variables Passed In

Dst : The destination for data output
elem_size : The size of a single element
num_elem : The number of elements in Src
fp : File pointer, file to read from

Local Variables

ans : the answer from “fread”
j : Byte count
Dst : Allows Dst to be treated as array of characters
temp : Temporary holder of a byte when swapping
ptr, ptr2 : Pointers to the two bytes to swap

Subroutine Calls (File:Subroutines)

None

revfwrite:

Return

size_t : Number of elements read

Author

Arthur Taylor, September 2002, MDL/RSIS

Arthur Taylor, November 2002, MDL <updated>

Purpose

To do an “fwrite”, but in a reverse manner

Note:

- 1) It is assumed that file is already open and in the correct place
- 2) Decided to write using a bunch of fput, since this is buffered. The thought here, is that it’s faster than swapping memory and then writing. This is the exact opposite method as revfread.

Variables Passed In

Dst : The destination for data output

elem_size : The size of a single element

num_elem : The number of elements in Src

fp : File pointer to write to

Local Variables

ans : the answer from “fread”

j : Byte count

Dst : Allows Dst to be treated as array of characters

temp : Temporary holder of a byte when swapping

ptr, ptr2 : Pointers to the two bytes to swap

Subroutine Calls (File:Subroutines)

None

Return

size_t : Number of elements read

memBitRead:

Author

Arthur Taylor, April 2003, MDL/RSIS

ATT, May 2003 <updated>

Purpose

To read bits from an unsigned char buffer array of memory. Assumes BufLoc is valid before first call. Typically this means do a bufLoc = 8” before the first call

Note:

- 1) Assumes binary bit stream is “big endian”. Resulting in no byte boundaries ie 00100110101101 => 00101101

Variables Passed In

Dst : Where to write results
 dstLen : Length in bytes of Dst
 Src : The data to read bits from
 numBits : How many bits to read
 BufLoc : Which bit to start reading from in Src.
 Starts at 8 goes to 1.
 numUsed : How many bytes from Src were used while reading

Local Variables

src : Allows Src to be treated as array of chars
 dst : Allows Dst to be treated as array of chars
 numBytes : How many bytes are needed in dst
 dstLoc : Where we are writing to in dst
 ptr : current byte we are writing to in dst
 BitMask : TBD

Subroutine Calls (File:Subroutines)

tendian.c: revmemcpy/memcpy (MEMCPY_BIG)

Return

int : 1 for error, 0 for success

memBitWrite:

Author

Arthur Taylor, April 2003, MDL/RSIS

Purpose

To write bits from a data structure to an unsigned char buffer array of memory. Assumes that the part of Dst we don’t write to have been correctly initialized. Typically this means do a “memset (dst, 0, sizeof(dst));” before the first call. Also assumes BufLoc is valid before first call. Typically this means do a “bufLoc = 8;” before the first call.

Note:

- 1) Assumes binary bit stream should be “big endian”
Resulting in no byte boundaries ie 00100110101101 =>
001001 | 1010110
- 2) Assumes that Dst is already zero’ed out

Variables Passed In

Src : The data to read from
srcLen : Length in bytes of Src
Dst : the char buffer to write the bits to
numBits : How many bits to write
BufLoc : Which bit is Dst to start writing to. Starts at 8 goes to 1

Local Variables

src : Allows Src to be treated as array of chars
dst : Allows Dst to be treated as array of chars
numBytes : How many bytes are needed in dst
srcLoc : Which bit we are reading from (location) in src
BitMask : TBD

Subroutine Calls (File:Subroutines)

tendian.c: revmemcpy/memcpy (MEMCPY_BIG)

Return

int : 1 for error, 0 for success

fileBitRead:

Author

Arthur Taylor, April 2003, MDL/RSIS

Purpose

To get bits from the file. Stores the current byte, and passes the bits that were requested to the user. Leftover bits, are stored in a gbuf, which should be passed in for future reads. If numBits == 0, then flush the gbuf

Variables Passed In

Dst : The storage place for the data read from file
dstLen : The size of dst in bytes
num_bits : The number of bits to read from the file
fp : The open file to read from
gbuf : The current bit buffer
gbufLoc : Pointer location of current position in bit buffer

Local Variables

BitRay : TBD
 buf_loc : TBD
 buf : TBD
 ptr : TBD

Subroutine Calls (File:Subroutines)

None

Return

int : 1 for error, 0 for success

fileBitRead:

Author

Arthur Taylor, August 2004, MDL/RSIS

Purpose

To write bits from src out to file. First writes out any leftover bits in pbuf, then bits from src. Any leftover bits that aren't on a full byte boundary, are stored in pbuf. If numBits == 0, then flush pbuf.

Variables Passed In

Src : The data to write to file
 srcLen : Length in bytes of src
 numBits : The number of bits to write to file
 fp : The opened file ptr to write to
 pbuf : the extra bit buffer
 pBufLoc : The location in the bit buffer

Local Variables

buf_loc : TBD
 buf : TBD
 ptr : TBD
 src : Pointer to SRC
 num_bytes : TBD
 src_loc : Pointer of current location in src data

Subroutine Calls (File:Subroutines)

tendian.c: revmemcpy/memcpy (MEMCPY_BIG)

Return

int : 1 for error, 0 for success

fileBitRead:

Author

Arthur Taylor, August 2004, MDL/RSIS

Purpose

To write bits from src out to file. First writes out any leftover bits in pbuf, then bits from src. Any leftover bits that aren't on a full byte boundary, are stored in pbuf. If numBits == 0, then flush pbuf.

Variables Passed In

Src : The data to write to file
 srcLen : Length in bytes of src
 numBits : The number of bits to write to file
 fp : The opened file ptr to write to
 pbuf : the extra bit buffer
 pBufLoc : The location in the bit buffer

Local Variables

buf_loc : TBD
 buf : TBD
 ptr : TBD
 src : Pointer to SRC
 num_bytes : TBD
 src_loc : Pointer of current location in src data

Subroutine Calls (File:Subroutines)

None

Return

int : 1 for error, 0 for success

3.4.10**Source file “tio3.c / tio3.h” Subroutine**

tOpen: (Exported)

Author

Arthur Taylor, August 2004, MDL/RSIS

Purpose

Opens new file

Variables Passed In

fid : The file ID associated with open file
 name : File name to open
 access : What kind of file access to open it with
 sys : Some endian 0 or opposite 1 as system.
 Alternatively...2 file made on intel, 3 file not made on intel.

Local Variables

ptr : Navigation pointer, init to head.

fp : file pointer to open file

Subroutine Calls (File:Subroutines)

None

Return

Null on error, or pointer to valid structure

tFind: (Exported)

Author

Unknown

Purpose

Returns a pointer to an already opened TIO_type given a file ID

Variables Passed In

fid : The file ID associated with open file

Local Variables

ptr : Navigation pointer, init to head.

Subroutine Calls (File:Subroutines)

None

Return

Null on error, or pointer to valid structure

tClose: (Exported)

Author

Unknown

Purpose

Close a TIO file, and free related data structures

Variables Passed In

tio : The opened file ptr to close

Local Variables

ptr : Navigation pointer, init to head

pt2 : Temporary navigation pointer

Subroutine Calls (File:Subroutines)

None

Return

1 if error, 0 if successful

tEndian_Switch (Exported)

Author
Unknown

Purpose
Toggles the endian'ness of a file

Variables Passed In
tio : The opened file ptr to adjust

Local Variables
None

Subroutine Calls (File:Subroutines)
None

Return
1 "true" if opposite endian, 0 "false" otherwise

tGet_Endian: (Exported)

Author
Unknown

Purpose
Returns a Boolean as to whether to treat the file as opposite endian to the operating system

Variables Passed In
tio : The opened file ptr to inquire from

Local Variables
None

Subroutine Calls (File:Subroutines)
None

Return
1 "true" if opposite endian, 0 "false" otherwise

tFlush: (Exported)

Author
Unknown

Purpose
Clears the bit buffer for tGetBit function

Variables Passed In
f_put : True, flush the put byte, false flush the get byte

tio : The opened file ptr to flush

Local Variables

None

Subroutine Calls (File:Subroutines)

None

Return

Number of bits put into the stream

tSeek: (Exported)

Author

Unknown

Purpose

To seek to specific locations in the file (see fseek)

Variables Passed In

tio : The opened file ptr to adjust
offset : distance from the origin
origin : SEEK_SET (0) from beginning
 SEEK_CUR (0) from current
 SEEK_END (2) from end

Local Variables

None

Subroutine Calls (File:Subroutines)

None

Return

0 if successful, non-zero on error

tGetBit: (Exported)

Author

Unknown

Purpose

To get bits (instead of bytes) from the file. Stores the current byte, and passes the bits that were requested to the user. Leftover bits, are stored in a char associated with the file, for future reads.

Variables Passed In

tio : The opened file ptr to read from
Dst : The storage place for the data read from
 file

dst_len : The size of dst (in bytes)
num_bits : The number of bits to read from the file

Local Variables

BitRay : TBD
buf_loc : TBD
buf : TBD
ptr : TBD
dst : Local pointer copy of Dst variable
num_bytes : TBD
dst_loc : TBD
c : TBD

Subroutine Calls (File:Subroutines)

None

Return

EOF if EOF, 1 if error, otherwise 0.

tRead: (Exported)

Author

Unknown

Purpose

To read into dst, which is an array of variables each of which is of elem_size. It does appropriate endian swapping as it reads

Variables Passed In

Dst : The storage place for the data read from file
elem_size : The size of a given element of Dst (bytes)
num_elem : The number of elements to read
tio : The opened file ptr to read from

Local Variables

ans : TBD
j : TBD
tmp : TBD
dst : TBD
ptr : TBD
ptr2 : TBD

Subroutine Calls (File:Subroutines)

None

Return

Number of elements read. Slows down to check for EOF

tReadStruct: (Exported)

Author

Unknown

Purpose

To read from file into dst which is a structure of some sort.
It also does appropriate endian swapping

Note:

- 1) specs of elem_sizes: example "1,4,2,8" => 1 char, 1 long, 1 short, 1 double Types larger than 9 (ascii 48..57) should be A..Z (ascii 65..90). (This is to reduce the calls to "atoi") "4*20," => 20 long ints. So you might have: "2,G,1,1*20,2". There is no "0*20" as the "same" endian machine wouldn't see it, so if I skipped either in dst or the file, it would only work on the other" endian machine.
- 1) It was felt that 1 call to fread with n/2 swaps is faster than n calls to fgetc.
- 2) Don't check tio->access, because user should be smart enough not to call this when inappropriate, and if they do, fread will put an error they can see by calling ferror.
- 3) If you have pointers, put them at the end, and decrease dst_len, or put them at the beginning and adjust both Dst and dst_len.
- 4) For 3 byte ints stored as longs, put them at front or back (same as pointers), then use tRead to fill them

Variables Passed In

Dst : The storage place for the data read from file
elem_size : The size of a given element of Dst (bytes)
dst_len : The length of the dst in bytes
tio : The opened file ptr to read from

Local Variables

ptr1 : TBD
temp : TBD
dst1 : TBD
dst2 : TBD
dst : TBD
array_size : TBD
j : TBD

Subroutine Calls (File:Subroutines)

None

Return

1 for true if an error occurred, otherwise 0 for success. See feof and ferror for EOF/errors

tPutBit: (Exported)

Author

Unknown

Purpose

To write bits from “src” out to a file. Leftover bits that aren’t on a full byte boundary are stored in buffer. Make sure to remember a flush buffer call when finished with the last PutBit call...call tFlush.

Note:

- 1) does not check tio->access
- 2) Don’t try to write more bits than you have
- 3) src is usually not an array of elements
- 4) Call tFlush (tio, 1) to flush the bit buffer

Variables Passed In

tio : The opened file ptr to read from
src : The data written to file
src_len : The size of a src (bytes)
num_bits : The number of bits to write to file

Local Variables

buf_loc : TBD
buf : TBD
ptr : TBD
src : Pointer to “Src” variable passed in
num_bytes : TBD

Subroutine Calls (File:Subroutines)

None

Return

Int : 1 error, 0 success

tWrite: (Exported)

Author

Unknown

Purpose

To write from “src” to file and “src” may be an array of variables of size “elem_size”. Endian swapping is not conducted during writing.

Note:

- 1) Arguments are ordered identically to fwrite command arguments
- 2) Example call:

```
int buff[100]
fwrite (&buff, sizeof (int), sizeof(buff)/sizeof(int), tio);
```
- 3) Couldn't do n/2 swaps and 1 call to fread because we would either corrupt the user's “src”, or need to call malloc. Instead use n calls to fgetc.
- 4) Don't check tio-access

Variables Passed In

Src : Data source to write to file
elem_size : Size of a given element of “src” (bytes)
num_elem : Number of elements to write
tio : The opened file “ptr” to write to

Local Variables

buf_loc : TBD
buf : TBD
ptr : TBD
src : Pointer to “Src” variable passed in
num_bytes : TBD

Subroutine Calls (File:Subroutines)

None

Return

size_t : number of elements written. (see feof and ferror for EOF/errors)

f2c Calls:

Author

Arthur Taylor, October, 3 1997 MDL

Purpose

A series of I/O definitions enabling “base SLOSH model” FORTRAN code access through f2c to the exported C procedures defined in tio3.c module

Subroutine Calls (File:Subroutines)

tio3.c: tFind, tRead, tWrite, tOpen, tClose

HP Calls:

Author

Arthur Taylor, October, 3 1997 MDL

Purpose

A series of I/O definitions enabling “base SLOSH model”
FORTRAN code access through f2c to the exported C
procedures defined in tio3.c module...FOR HP SYSTEMS.

Subroutine Calls (File:Subroutines)

tio3.c: tFind, tRead, tWrite, tOpen, tClose

4.0 EXECUTING THE SLOSH MODEL

4.1 Execution Commands for SLOSH

To execute the SLOSH model from the command line use the following:

<u>slosh3</u>	<u>hchs</u>	<u>./sloshbsn</u>	<u>hugo.trk</u>	<u>hugo.rex</u>	<u>hugo.env</u>
1	2	3	4	5	6

1. Compiled SLOSH model executable
2. Desired/selected basin – see figure 6
3. Location of basin data file
4. TRK input file - section 4.2
5. REX output file
6. Envelop of high water output file

4.2 Creating a TRK Data File

The SLOSH TRK file is data input that contains 100 track points of data as broken down below – see figure 32.

Header:

1. Must have 2 lines for program, however free form, actual content not used by program.
2. Storm name & forecaster initials : HUR HUGO BEST TRK BY BRJ
3. Datums : DATUMS = 2.1/2.1 FT
4. Forecast pressure change : DELTA-P = 76MB
5. Saffir-Simpson Category : CAT4
6. Radius of maximum winds (statute miles): RMW = 13 ST MI
7. Storm motion : NW/28MPH

Data: (100 data points - 8 colums)

1. Col 1 - Track point 1-100
2. Col 2 - Latitude
3. Col 3 - Longitude

4. Col 4 - Wind speed (mph)
5. Col 5 - Storm direction & speed (knots)
6. Col 6 - Delta-P (mb)
7. Col 7 – RMW (mph)
8. Col 8 – Actual STM points for human reference not used by program
9. NAP = Nearest Approach Point (ie land fall, typically at point #70)

Footer:

1. Three line footer necessary, format specific, used by SLOSH program
2. Line 1: 59 76 70
 - 59 = IBGNT = Beginning hour, starting point, 59th point in the 100pt track file
 - 76 = INTEND = End hour, end point, 76th point in the 100 pt track file
 - 70 = JHR = Near approach / landfall point. Typically 70th point in the 100 pt track file
3. Model run GMT: HR0000 22 SEP 1989
4. 2.1 2.1 – Sea and lake datum in feet. Initial water level height

Header									
HUR H U G O BEST TRK BY BRJ; DATUMS= 2.1/2.1 FT									
DELTA-P = 76MB ; CAT 4 ; RMW= 13 ST. MI.; NW/28MPH									
									Data
1	22.2287	67.578	10.44	313.13	60.00	30.00			1
2	22.3321	67.697	10.44	313.13	60.00	30.00			2
3	22.4355	67.816	10.44	313.13	60.00	30.00			3
4	22.5389	67.936	10.44	313.13	60.00	30.00			4
5	22.6423	68.055	10.44	313.13	60.00	30.00			5
6	22.7457	68.175	10.44	313.13	60.00	30.00			6
7	22.8490	68.294	10.44	313.13	60.00	30.00			7
8	22.9524	68.414	10.44	313.13	60.00	30.00			8
9	23.0558	68.534	10.44	313.13	60.00	30.00			9
NAP	---	10 23.1592	68.654	10.44	313.13	60.00	30.00	10	---NAP
.....SNIP.....									
95	46.5284	73.246	70.10	58.18	20.00	20.00			1
96	47.0566	71.980	70.10	58.18	20.00	20.00			2
97	47.5848	70.701	70.10	58.18	20.00	20.00			3
98	48.1128	69.410	70.10	58.18	20.00	20.00			4
99	48.6406	68.104	70.10	58.18	20.00	20.00			5
100	49.1683	66.785	70.10	58.18	20.00	20.00			6
59 76 70 IBGNT ITEND JHR									
HR0000 22 SEP 1989 NEAREST APPROACH, OR LANDFALL, TIME									
2.1 2.1 SEA AND LAKE DATUM									
Footer									

Figure 32 – TRK Data File Breakdown

5.0 SLOSH DISPLAY PROGRAM GRAPHICAL USER INTERFACE

This section breaks down each component of the SLOSH Display Program's Graphical User Interface (GUI). When first launched the initial state of the SLOSH Display will either be the most recently selected SLOSH basin or a map of the Washington D.C. region – Figure 33.

The SLOSH Display Program menu bar options are as follows, label 1, Figure 33:

Menu Bar Option: File:

Save to PCX: Saves SLOSH Display as a .pcx graphic image.

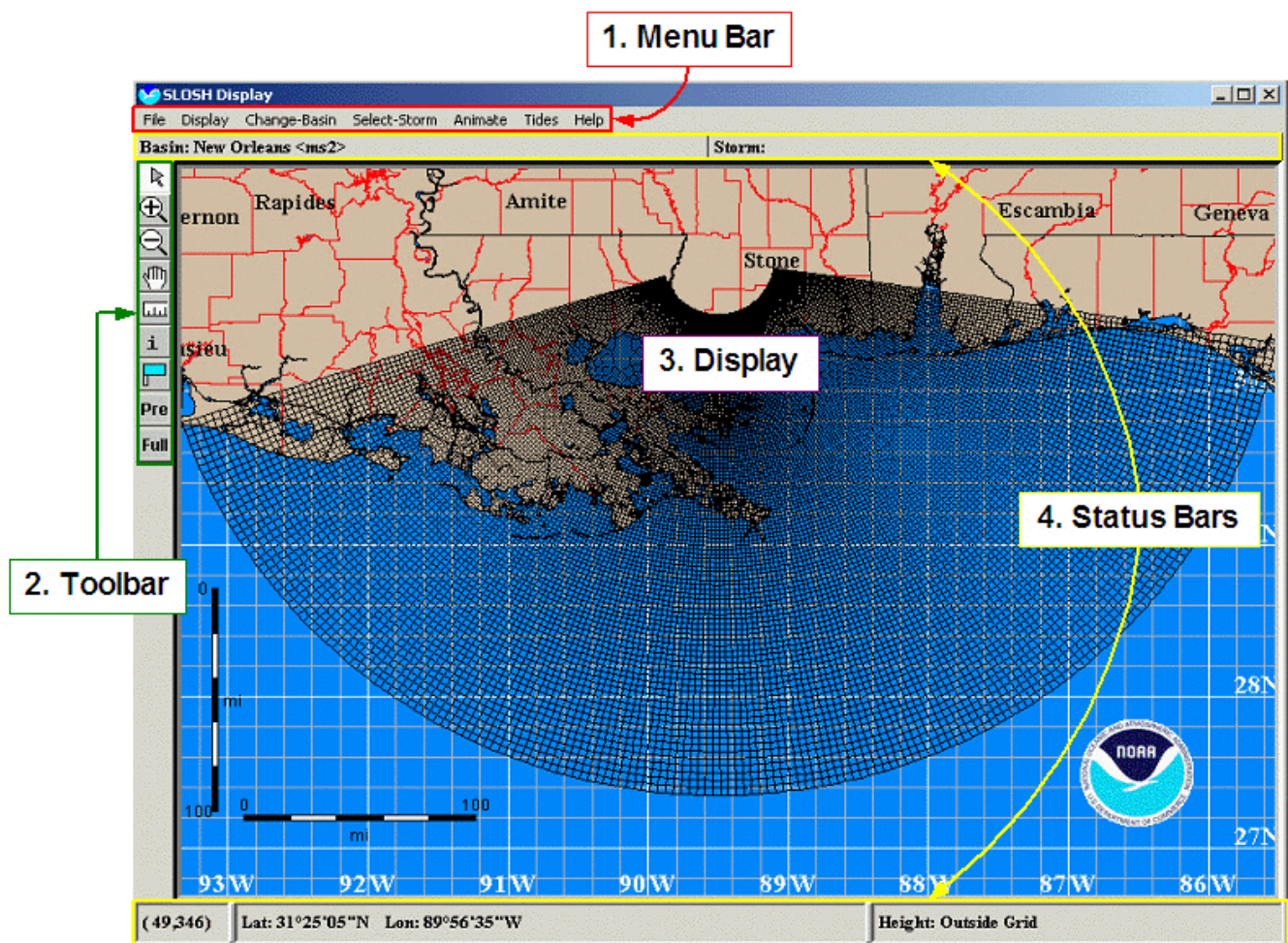


Figure 33 – SLOSH Display Program Initial Graphical User Interface

Save to TIF: Saves “SLOSH Display” as a graphical image, TIF format.

Update Basin Table: Function to ingest or update the SLOSH database (sloshdsp.ini) with new basins added post install. New .bnt (.bnt = basin name table) files are placed in the default directory c:\slosh.pkg\data and then this function is selected to import/update.

Update Data Root Directories: Function allowing users to edit default path information defining where SLOSH looks to ingest Rex and MEOW/MOM files and also identifies where output data will be stored.

Quit: Exit SLOSH Display Program

Menu Bar Option: Display:

Label Counties: Toggle to display counties on map background.

Label Locations: Toggle to display town/city names on map background.

Label Buoys: Toggle to display ocean buoys on map background.

Label Surge: Toggle to display surge data.

Tracks: Option to display hurricane tracks associated with an Envelop or a MEOW, the latitude and longitude storms pass through.

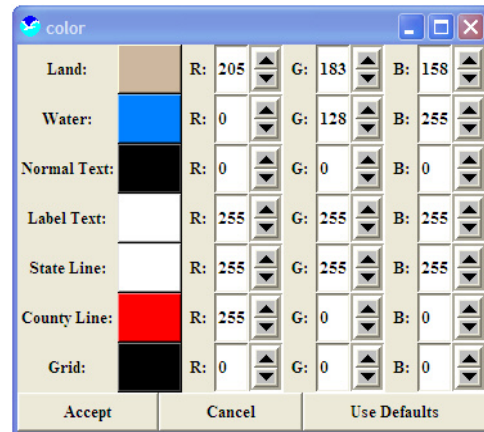


Figure 34 – Change Color GUI

Notes about tracks:

1. With MOMs there would be too many tracks to display so they are excluded. Rex file tracks are always on, however when pausing an animation and then clicking on Tracks option, the tracks and circle of maximum winds will be removed. Resuming the animation will redraw the tracks and wind.

SLOSH Grid: The SLOSH model uses a polar, elliptical, or hyperbolic grid for its computations, depending on the slosh basin. The reason for not using a Cartesian grid is for finer resolution near the shore.

Entire Grid: Toggle to display entire SLOSH storm surge grid. Default shortens the extreme outer grids

Fixed Color Scale: The default color scale for storm surge heights customizes itself for each storm and basin chosen. The scale can be modified the up and down arrow buttons located to the right of the color spectrum on the surge height legend. The Fixed Color Scale then enables the user to keep the color scale from changing when different storms are loaded, making it easier to visually compare storms.

Change Colors: This option opens a color selection window used to change the colors of the land, water, text coastline, or grid on the SLOSH Display. Colors can be modified by clicking the up and down arrows for red (R), green (G), and blue (B) – figure 34.

Cascading Menu: **Lat/Lon Grid:**

No Lat/Lon Grid: Toggle to display latitude and longitude overlay on SLOSH map background. Radio box with No Lat/Long Grid, Grid & SubGrid, Grid, and Lattice selections.

Grid & SubGrid: Displays SLOSH basin grid and Latitude/Longitude graph on SLOSH Display. Radio box with No Lat/Long Grid, Grid & SubGrid, Grid, and Lattice selections.

Grid: Displays only SLOSH basin grid on SLOSH Display. Radio box with No Lat/Long Grid, Grid & SubGrid, Grid, and Lattice selections.

Lattice: Toggle between Lat/Lon line grid to Lat/Lon corner marker grid. Radio box with No Lat/Long Grid, Grid & SubGrid, Grid, and Lattice selections.

Automatic: SLOSH software assigns latitude/longitude and sub grid spacing to default map background of selected image, one degree equals 60 nm. Toggle with *1 degree (.2 subgrid)*, *1 degree (1.25 subgrid)*, *5 degree blocks*, *10 degree blocks*, *15 degree blocks* functions.

1 degree (.2 subgrid): Sets latitude and longitude to 1 degree spacing and the graphical subgrid to .2 degrees on the map background. Toggle with *Automatic*, *1 degree (1.25 subgrid)*, *5 degree blocks*, *10 degree blocks*, *15 degree blocks* functions.

1 degree (1.25 subgrid): Sets latitude and longitude to 1 degree spacing and the graphical subgrid to 1.25 degrees on the map background. Toggle with *Automatic*, *1 degree (.2 subgrid)*, *5 degree blocks*, *10 degree blocks*, *15 degree blocks* functions.

5 degree blocks: Sets latitude and longitude to 5 degree spacing. Toggle with *Automatic*, *1 degree (.2 subgrid)*, *1 degree (1.25 subgrid)*, *10 degree blocks*, *15 degree blocks* functions.

10 degree blocks: Sets latitude and longitude to 10 degree spacing. Toggle with *Automatic*, *1 degree (.2 subgrid)*, *1 degree (1.25 subgrid)*, *5 degree blocks*, *15 degree blocks* functions.

15 degree blocks: Sets latitude and longitude to 15 degree spacing. Toggle with *Automatic*, *1 degree (.2 subgrid)*, *1 degree (1.25 subgrid)*, *5 degree blocks*, *10 degree blocks* functions.

Grid over land & water: Selection covers both land and water map regions with latitude/longitude and subgrid graphics. Toggle with *Grid over land* function.

Grid over land: Selection to display latitude/longitude and subgrid graphics over land only. Toggle with *Grid over land & water* function.

Cascading Menu: **Units:**

Surge (Feet): Toggle storm surge data in feet. Toggle with Surge (Meters) selection.

Surge (Meters): Toggle storm surge data in meters. Toggle with Surge (Feet) selection.

Distance (Statute Miles): Toggle calculated distance in Statute Miles. Toggle with Distance (Kilometers) and Distance (Nautical Miles) selections.

Distance (Kilometers): Toggle calculated distance in Kilometers. Toggle with Distance (Statute Miles) and Distance (Nautical Miles) selections.

Distance (Nautical Miles): Toggle calculated distance in nautical miles. Toggle with Distance (Statute Miles) and Distance (Kilometers).

Degrees (minutes/seconds): Toggle Lat/Lon status bar display (label 4, Figure 33) to minute/second format

Degrees (decimal): Toggle Lat/Long status bar display (label 4, Figure 33) to decimal format

Quad: Toggle degree in status bar to SLOSH grid quadrant (label 4, Figure 33)

Cascading Menu: **Zoom:**

None: (*arrow*) in the tool bar, turns all Zoom commands off and enables users to move the labels/map keys.

Zoom In: (+) in the tool bar, zoom into desired map location. Select function, left mouse click and release over map, drag box over desired area on map, left mouse click to zoom. Slight lag expected. Function must be reselected every time to repeat.

Zoom Out: (-) in the tool bar, zoom out map view. To use: Select function, left mouse click anyplace over map. Slight lag expected. To repeat simply left click mouse button over map until desired map perspective.

Pan: (*hand*) in the tool bar, function to reset the map center point. To use: Select function, left mouse click and hold, move pointer as desired, release mouse button.

Ruler: (*ruler*) in the tool bar, function to measure distance between desired points. To use: Select function, left mouse click and release over first location point, move cursor to next point, left mouse click, and repeat as desired. Top measurement reflects most recent point-to-point measurement and the bottom is the total measurement from most recent point to last point. To remove line measurements click right mouse button.

Inquire All: (*i*) in the tool bar, functionality to display MEOW graphics on the SLOSH map background also providing users a method to interrogate MEOW on a individual cell basis. For more detail about *Inquire All* reference section 5.3

Probe: (*flag*) in the tool bar, determines height of storm surge by placing “measurement” flags in desired locations. Not intended for overland use. To use: Select function, left mouse click over desired “water” location, repeat placement as desired.

Previous Zoom: (*pre*) in the tool bar, enables the user to go back to the last zoom level, all the way back when the SLOSH basin was last changed.

Full Zoom: (*Full*) in the tool bar, enables the user to redraw the SLOSH basin in its original size.

Cascading Menu: View:

Full Window: Restores the SLOSH Display to it’s original position.

Window A, B, C: Selections to enlarge the SLOSH display to predefined zoom levels for selected basin.

Discrete Color Scale: Allows users to display the storm surge using one color for a set of values. Usually this is one color for (0,1] feet, another for (1,2] feet (or 0, 1] meter).

Value Follow Mouse: Displays storm surge values at the cursor location. Values are also displayed at the bottom of the screen in the Height Label – label 4, Figure 33.

Menu Bar: Change-Basin:

Change-Basin: Change the selected basin view. To switch basins double click on the desired basin from the presented list – Figure 35.

Cascading Menu: Which Coast?:

Atlantic: While in the *Change-Basin* mode as shown in figure 35 selecting *Atlantic* will switch the map background to an Atlantic view.

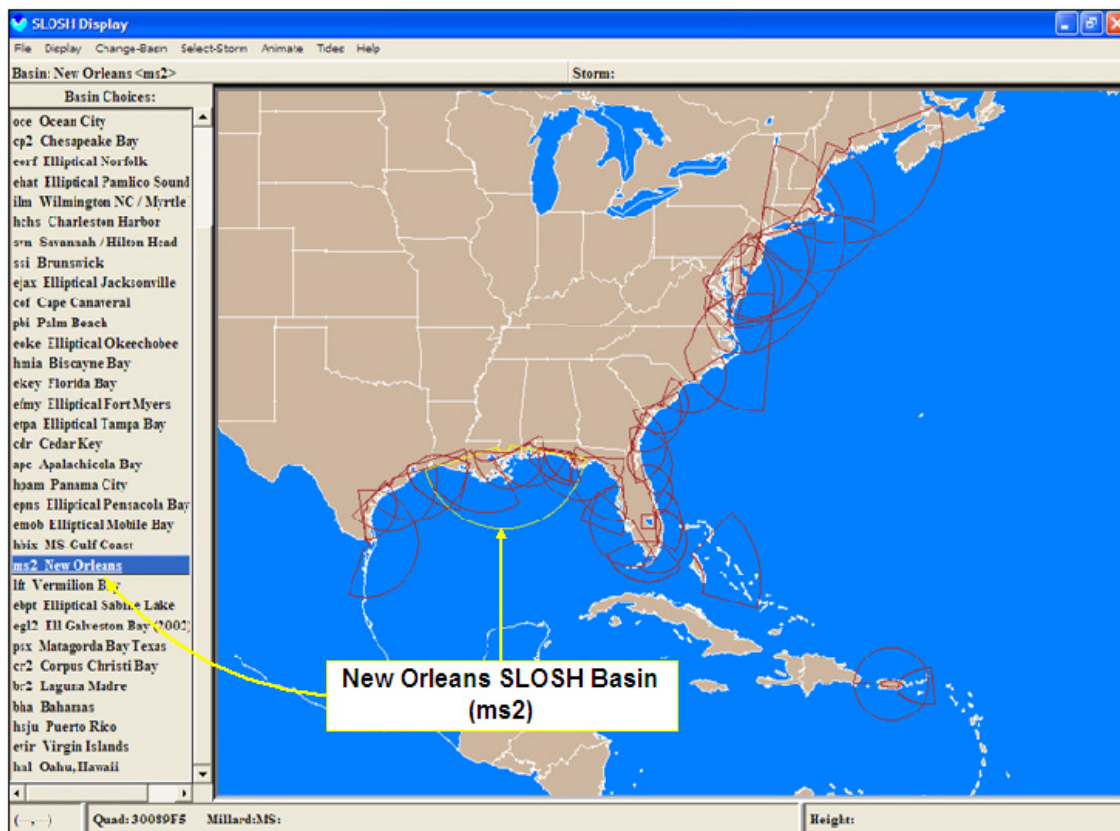


Figure 35 – Change-Basin Graphical User Interface

Pacific: While in the *Change-Basin* mode as shown in figure 35 selecting *Pacific* will switch the map background to a Pacific view.

Cascading Menu: **Which Basins?**

Installed: While in the *Change-Basin* mode as shown in figure 35 selecting *Installed* will displayed all default basins originally installed.

All: While in the *Change-Basin* mode as shown in figure 35 selecting *All* will display every basin recognized by SLOSH, including installed defaults and post-installed basins.

Menu Bar: **Select Storm:**

Select Storm: Functions to simulate various storm surge measurements (Historical, MEOW, MOM) reference 5.2 for full detail.

Inquire All: Functionality to display MEOW graphics on the SLOSH map background also providing users a method to interrogate MEOW on a individual cell basis. For more detail about *Inquire All* reference section 5.3.

Menu Bar: **Animate:**

Animate .rex File: Time-lapsed animated “movies” of historical SLOSH based storm surge data, representing actual storm surge water level information. For instructions to use this function reference section 5.4.

Import .rex File: Import a .rex file SLOSH Display Program. New files are copied into c:\slosh.pkg\data\rexfiles – section 5.5.

No Wind: Check box toggling wind barbs display on or off of rex image.

Wind (follow track): Check button to sets wind barbs to follow selected storm loop

Wind (fixed points): Check button to sets wind barbs to remain stationary to map background as storm loops

Pressure Contours: Check button to display pressure contours around selected storm.

Wind Grid (l x w): Sets the size (the square, length x width) of the wind grid around a storm, default is 21x21.

Anim to .pcx files: Function allowing users to generate and export animated GIF images.

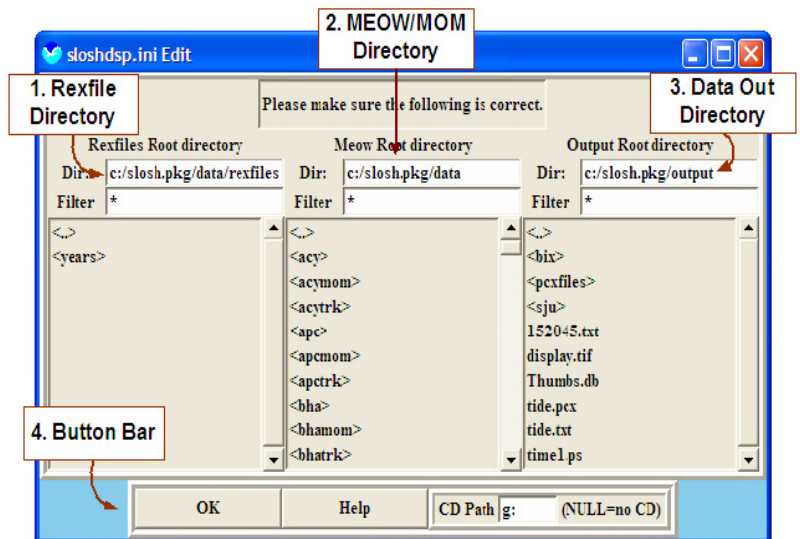


Figure 36 – sloshdsp.ini Edit User Interface

5.1 SLOSH sloshdsp.ini Edit Graphical User Interface

The *sloshdsp.ini* file provides path and other configuration parameters used by the SLOSH Display program. The sloshdsp.ini GUI allows users an interface to edit the most commonly used paths for: 1) rex files, 2) MEOW/MOM files, and 3) output data. The default paths are c:\slosh.pk\data\rexfiles, c:\slosh.pk\data, and c:\slosh.pkg\data\ respectively as set during the install process.

The *sloshdsp.ini Edit* interface breaks down very simply as shown in figure 36:

1) Rexfile directory: default is c:\slosh.pkg\data\rexfiles. Users can change this path by navigating to the desired directory from the “Rexfiles Root directory” directory list box.

2) Meow/MOM Directory: default is c:\slosh\pkg\data. Users can change this path by navigating to the desired directory from the “Meow Root directory” as shown. Note: Although MOMs are not described they will also be associated with this listing/directory.

3) Data Out Directory: default is c:\slosh.pkg\output. Users can change this path by navigating to the desired directory from the “Output Root directory”

4) Button Bar: There are three components to the button bar:

- OK – Will save and close the *sloshdsp.ini Edit* interface
- Help – Launches help interface providing instruction to the *sloshdsp.ini Edit* interface, similar to this section.
- CD Path – Defines the CD-ROM drive with the initial default as “g:”. If no CD-ROM exists users will receive a pop-up window to define a path on the local drive to retrieve data.

5.2 Storm Selection

From the SLOSH Display program’s menu bar option “Select Storm -> Select Storm” a separate user interface is launched allowing users to simulate storm surge related measurements from Historical, MEOW, and MOM based perspectives.

5.2.1 Display Historical SLOSH Model Data

The SLOSH Display program provides an option where users can display historical storm data but only for the following seven storms. To use the historical display:

Basin	Storm File
Apalachicola Bay	agnes.apc
Bahamas	andrwnhc.bha
Sabine Lak	audrey.ebp
Fort Myers	andrew.fmy
Virgin Island	h1928.vir
Oahu, Hawaii	oahu1.hnl
Puerto Rico	h1928.sju

Figure 37 – SLOSH Historical Storm Data Chart

- 1) In the SLOSH Display Program select desired basin – see figure 35.
 - ▶ Menu Bar: **Change-Basin -> Change Basin**
 - ▶ **In the *Change Basin* interface double click on the desired based in list box**
- 2) Launch the *Select Storm* user interface.
 - ▶ Menu Bar: **Select Storm -> Select Storm**
- 3) Select the Historical option of the *Select Storm* interface – label 1, figure 38
 - ▶ **Select Historical radio button, top center**

4) Select desired Historical storm from list – label 3, figure 38:

► **Select from list box as shown**

Note: Historical storm file naming breaks down as follows

- *E10510.ms2:*
 - *E = direction: east storm motion*
 - *1 = Category: Cat 1-5*
 - *05 = Forward Speed: mph*
 - *10 = Mean or High Tide: Number other than 0 is high tide (EG., I2 means high tide of 2ft NGVD29 or I0 means mean tide of 0ft NGVD29)*
 - *.ms2 = basin*

5) Generate Historical display – see figure 38:

► **Select Apply Button**

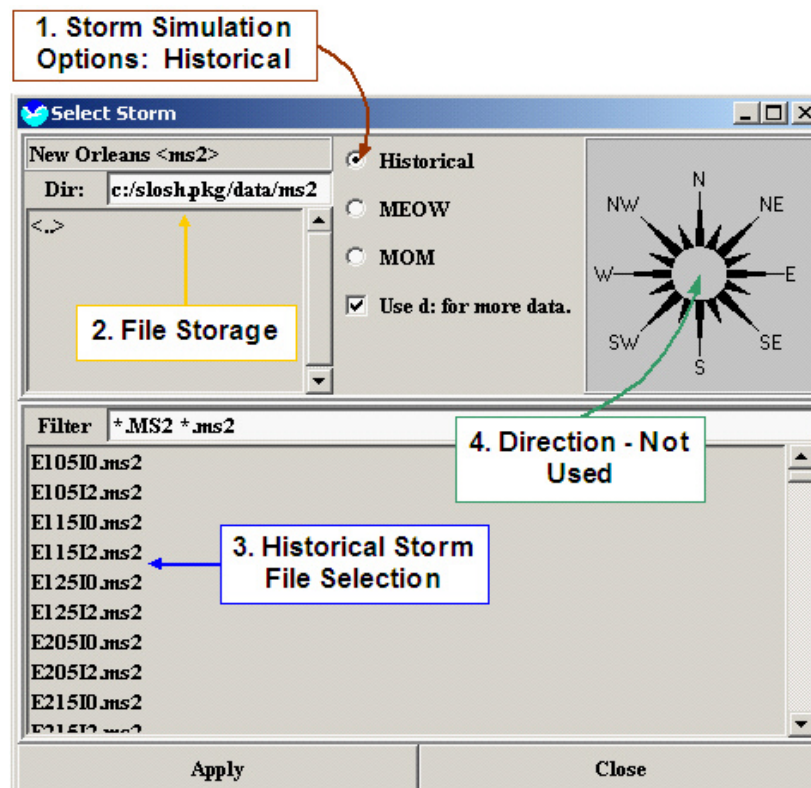


Figure 38 – Select Storm “Historical” Interface

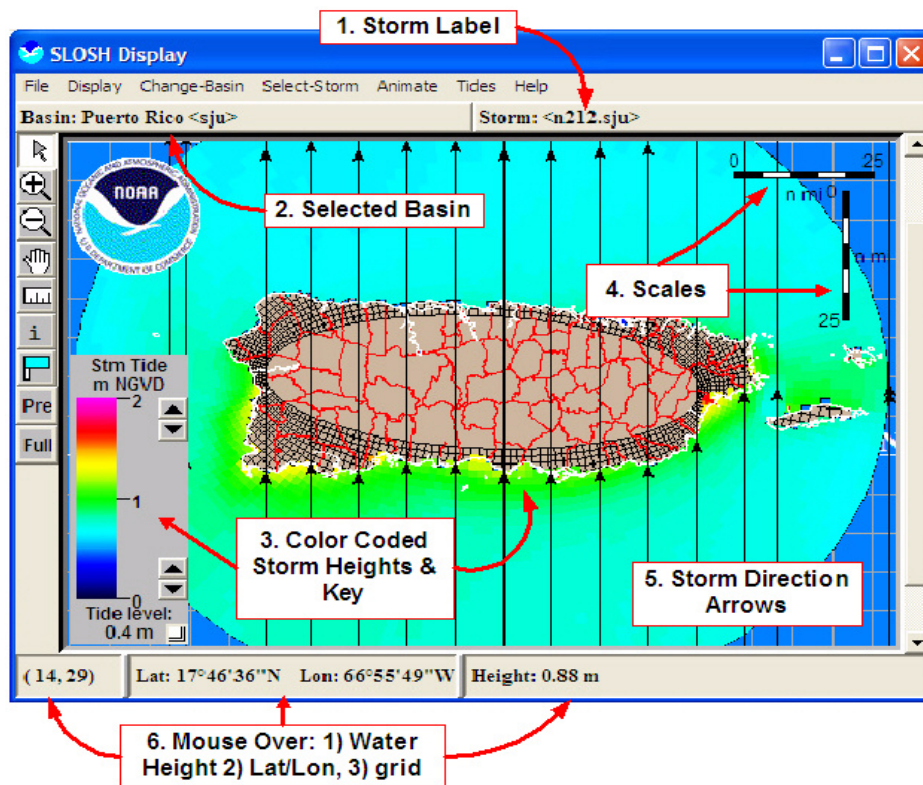


Figure 39 – Selected Storm “Historical” Display

5.2.2 Display MEOW SLOSH Model Data

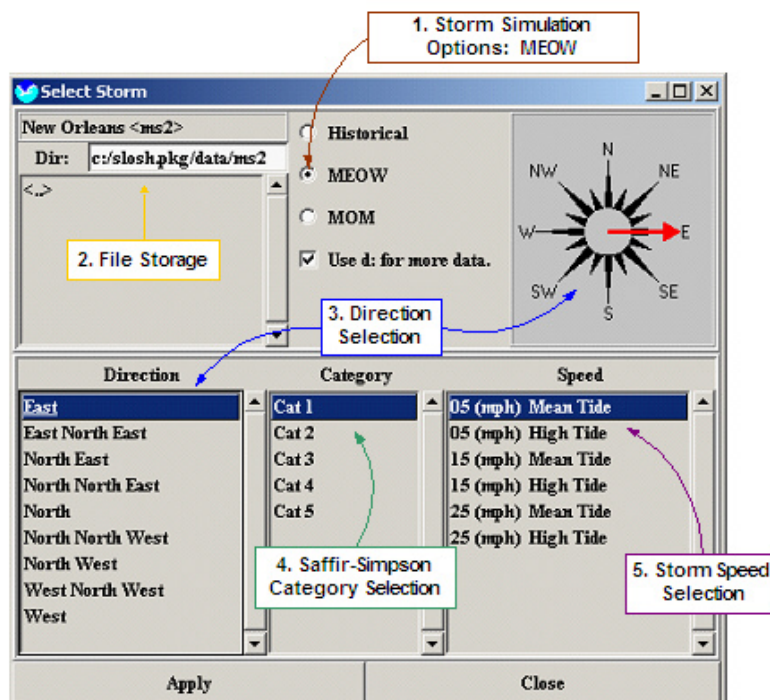


Figure 40 – Select Storm “MEOW” Interface

The *Maximum Envelope of Water* (MEOW) is a composite of maximum storm surge heights at each SLOSH grid cell using hypothetical hurricane runs with the same: 1) category, 2) forward speed, 3) landfall direction, 4) initial tide level, and 5) radius of maximum winds. Users can view MEOWs in two ways: 1) Using the Inquire tool described in section 5.3, or 2) through the Select-Storm Interface steps described in this section.

- 1) In the SLOSH Display Program select desired basin – see figure 35.
 - ▶ Menu Bar: **Change-Basin -> Change Basin**
 - ▶ **In the *Change Basin* interface double click on the desired basin in list box**
- 2) Launch the *Select Storm* user interface.
 - ▶ Menu Bar: **Select Storm -> Select Storm**
- 3) Open the MEOW option of the *Select Storm* interface – label 1, figure 40:
 - ▶ **Select MEOW radio button, top center**
- 4) Select desired storm motion (from) – label 3, figure 40:
 - ▶ **Use list box selection or compass as shown**
- 5) Select storm category/strength (Saffir-Simpson) – label 4, figure 40:
 - ▶ **Select category 1-5 as desired from list box**
- 6) Select desired storm speed and tide information – label 5, figure 40:
 - ▶ **Select from list box as shown**
 - Note: 1) Selecting high tide option is recommended to generate a conservative estimate as high tide selections create greater storm surge.*
- 7) Generate MEOW Image:
 - ▶ **Select Apply Button**

After generating the MEOW in the SLOSH display it should look similar to figure 42. The storm direction arrows represent each hypothetical hurricane track used to generate the MEOW. The arrow indicates storm movement, the category and direction are constant and the MEOW is generated by taking, in each grid cell, the maximum of the hypothetical storm surge/storm tide envelopes.

5.2.3 Display MOM SLOSH Model Data

The Maximum of MEOWs (MOM) is a composite of the maximum storm surge/storm tide for all hurricanes of a given category. Only one storm per category or five MOMs per basin are produced which disregards forward speed, landfall direction and landfall locations. The following instructions will assist users in displaying a MOM for any desired basin:

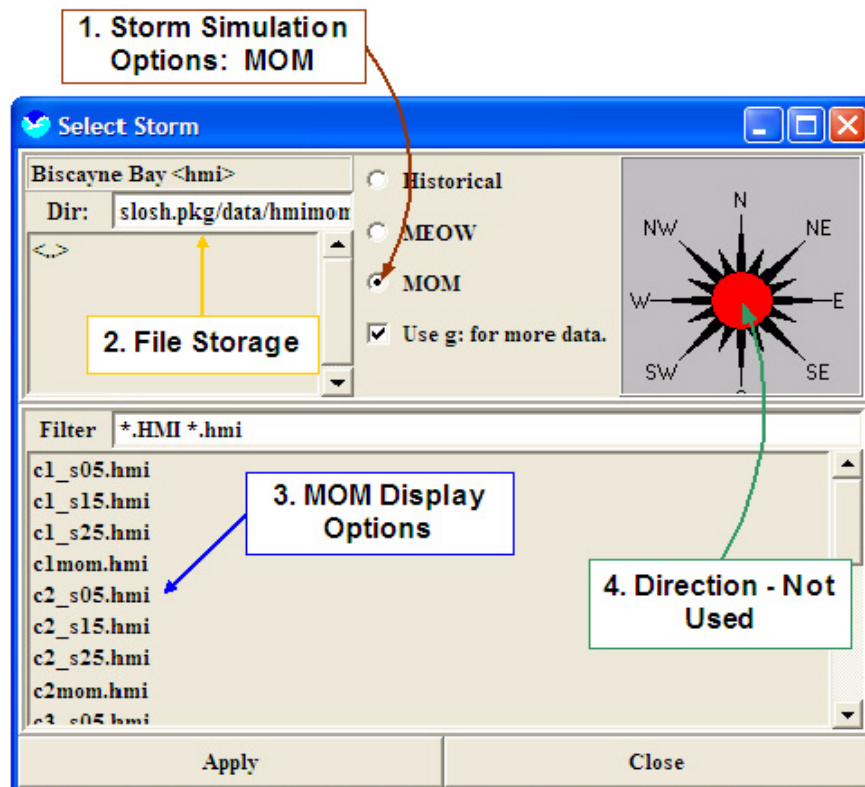


Figure 41 – Select Storm “MOM” Interface

- 1) In the SLOSH Display Program select desired basin – see figure 35
 - ▶ Menu Bar: **Change-Basin -> Change Basin**
 - ▶ In the *Change Basin* interface double click on the desired basin in list box
- 2) Launch the *Select Storm* user interface.
 - ▶ Menu Bar: **Select Storm -> Select Storm**
- 3) Open the MOM option of the *Select Storm* interface – label 1, figure 41
 - ▶ **Select MOM radio button, top center**
- 4) Select the desired filename in the *Select Storm* interface – label 1, figure 41
 - ▶ **Select from file list box as shown**

Note: 1) File naming conventions vary between basins, however each name will follow similar naming conventions as the examples below:

- *c2_s25.hmi* = Cat 2, Direction South at 25 kts, Basin hmi
- *H11_MOM.MS2* = High Tide, Cat 1, MOM, Basin MS2
- *C5_MEAN.MS2* = Cat 5, Mean Tide, Basin MS2

2) For offices from Pamlico Sound south, MOMs are listed as fast, slow, or mom; the Norfolk basin to the Atlantic City basin MOMs are listed as mean or HIx – where x is the category number and HI means high tide; and the New York City basin to the Pemobscot basin are listed as mean (me or mean) or high (hi) along with a category number.

5) Generate MOM Image – see figure 42

► **Select Apply Button**

The only obvious difference between MOM and MEOW output is the storm label as shown in figure 42, label 1. Notice the absence of storm direction arrows? Since the MOM is the maximum of the MEOWs with no direction associated it is therefore missing in the display

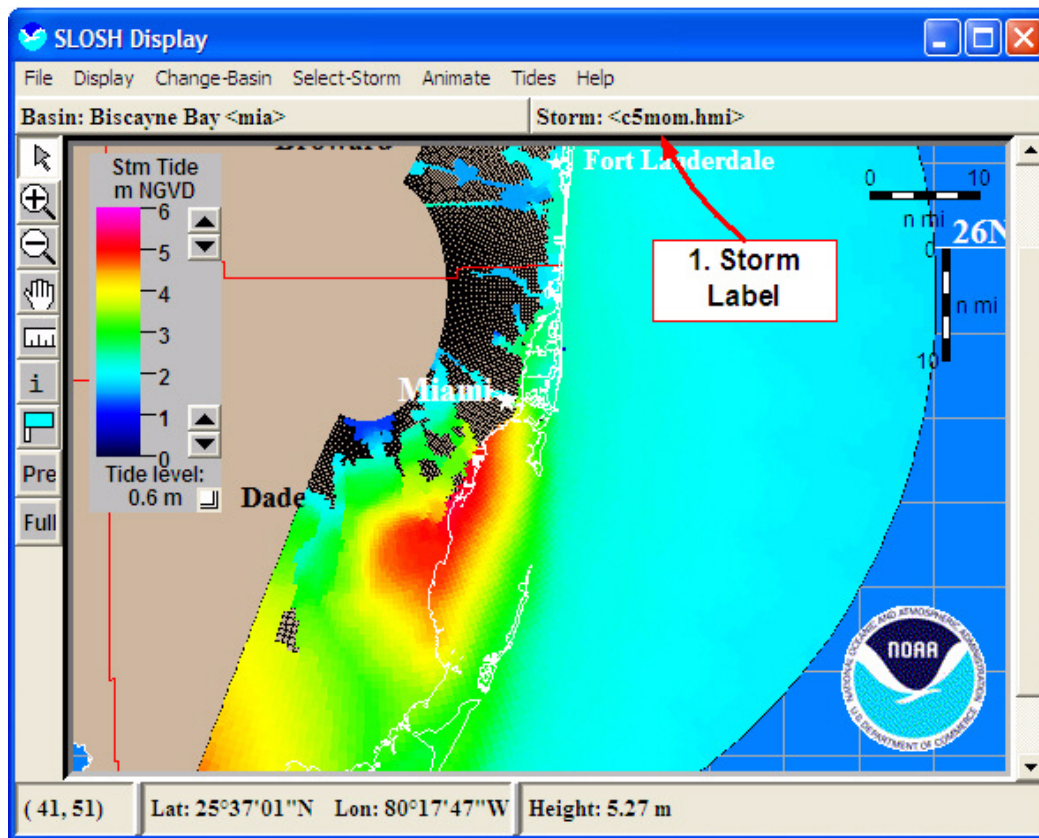


Figure 42 – Example Cat 5 MOM Display for Biscayne Bay Basin

5.3 Inquire All

The SLOSH Display Program's *Inquire All* functionality allows users to display all available Maximum Envelope of Water (MEOWs) images, similar to the *Selected Storm MEOW* method found in section 52.2. What makes this functionality unique is that individual cells that comprise a MEOW image can be interrogated independently. This data then shows how different combinations of storm category, wind speed, and forward speed affects the maximum potential storm surge/storm tide of the selected location.

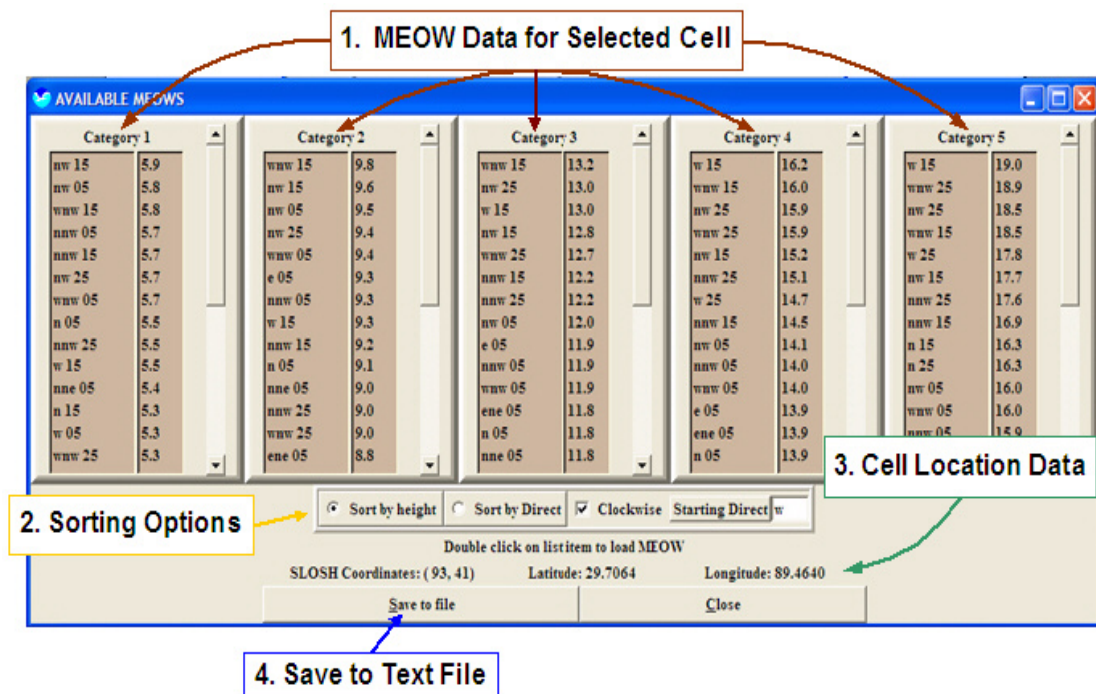


Figure 43 – Inquire All User Interface

The *Inquire All* user interface breaks down into 4 major components as shown in figure 43. Label one represents five Saffir-Simpson category list boxes providing users detailed MEOW data of the selected cell based on direction, speed, and category. Users can also graphically display related list box data by double clicking on the desired field which is then displayed on the SLOSH map background. To ease searching for the desired MEOW data list sorting functions are provided allowing users to sort list box data by height and direction as shown in label 2. Label 3 highlights the cell location by grid reference and Latitude/Longitude. Finally label 4 points out the *Save to File* option to save grid data in a text file – see figure 44. When using the Save to File function note that files will be stored in c:\slosh.pkg\output\<basin id> and by grid number ending in .hgt unless otherwise changed. Below are steps to use the *Inquire All* Functionality.

- 1) In the SLOSH Display Program select desired basin – see figure 35:
 - ▶ Menu Bar: **Change-Basin -> Change Basin**
 - ▶ In the *Change Basin* interface double click on the desired based in list box
- 2) Select the desired grid to analyze:
 - ▶ Menu Bar: **Display -> Zoom -> Inquire All**
 - ▶ Place “question mark” cursor over grid to analyze
 - ▶ Over desired grid select with left mouse button

Note: May be necessary to zoom to select desired grid
- 3) In the launched Inquire All interface expand window, view and/or save data.

Note: 1) To view graphical image of desired data point in list box, double click with left mouse button. 2) saved files will be stored in

c:\slosh.pkg\output*<basin id>* and by grid number ending in .hgt unless otherwise changed.

- 4) Close *Inquire All* interface
 - **Select close button**

Surge heights at a given point for all available MEOVS								
Basin = sju								
SLOSH grid reference (I,J) = (4,146)								
Approximate Lat = 18.503184 Long = 66.778931								
Category 1			Category 2			Category 3		
dir	sp	surge	dir	sp	surge	dir	sp	surge
n	12	1.8	n	12	2.4	n	12	3.0
nne	12	1.8	nne	12	2.4	nne	12	3.0
ne	12	1.7	ne	12	2.3	ne	12	2.9
nnw	12	1.7	nnw	12	2.3	nnw	12	2.9
nw	12	1.7	nw	12	2.3	nw	12	2.8
w	12	1.7	w	12	2.2	w	12	2.8
wnw	12	1.7	wnw	12	2.2	wnw	12	2.8
ws	12	1.6	ws	12	2.0	ws	12	2.5
Category 4			Category 5			Historical		
dir	sp	surge	dir	sp	surge	dir	sp	surge
n	12	3.6	n	12	4.2	h1928		1.8
nne	12	3.6	nne	12	4.2			
ne	12	3.5	ne	12	4.1			
nnw	12	3.5	nnw	12	4.1			
nw	12	3.4	nw	12	4.0			
w	12	3.3	w	12	3.9			
wnw	12	3.3	wnw	12	3.8			
ws	12	2.9	ws	12	3.4			

Figure 44 – Inquire All, Grid Specific, Text Output

5.4 Animate Historical Rex Data

This section describes the necessary features and functionality to display and animate Rex data in the SLOSH Display program. The provided historical Rex data files delivered with the SLOSH CDrom are as follows:

Rex Filename	Basin Name	Rex Filename	Basin Name
bret.rex	cr2: Corpus Christi Bay	eloise.rex	epns: Elliptical Pensacola Bay
dennis_1.rex	ehat: Elliptical Pamlico Sound	agnes.rex	apc: Apalachicola Bay
dennis_2.rex	ehat: Elliptical Pamlico Sound	camilbix.rex	hbix: MS-Gulf Coast
floydbha.rex	bha: Bahamas	camilms2.rex	ms2: New Orleans
floydchp.rex	cp2: Chesapeake Bay	betsy.rex	ms2: New Orleans
floydeht.rex	ehat: Elliptical Pamlico Sound	carla.rex	psx: Matagorda Bay Texas
floydilm.rex	ilm: Wilmington NC / Myrtle Beach	donnaemy.rex	efmy: Elliptical Fort Myers
bonnie.rex	ehat: Elliptical Pamlico Sound	gracie.rex	hchs: Charleston Harbor
earl.rex	apc: Apalachicola Bay	audrey.rex	ebpt: Elliptical Sabine Lake
geor_mob.rex	emob: Elliptical Mobile Bay	hazel.rex	ilm: Wilmington NC / Myrtle Beach
geor_ms2.rex	ms2: New Orleans	oke1949.rex	eoke: Elliptical Okeechobee
Georg_la.rex	hbix: MS-Gulf Coast	ms2-1947.rex	ms2: New Orleans
Georg_sj.rex	hsju: Puerto Rico	ms2-1947.rex	ms2: New Orleans
Georg_vi.rex	evir: Virgin Islands	bos1944.rex	box: Boston Harbor
ERIN.rex	cof: Cape Canaveral	nyc1944.rex	ny2: New York
opal.rex	epns: Elliptical Pensacola Bay	pvd1938.rex	pvd: Narragansett/Buzzard Bays
andr_ms2.rex	ms2: New Orleans	chp1033.rex	cp2: Chesapeake Bay
andrew_b.rex	bha: Bahamas	sju1928.rex	hsju: Puerto Rico
andrewfl.rex	hmia: Biscayne Bay	eky1926	ekey: Florida Bay
andrewla.rex	lft: Vermilion Bay	mia1926.rex	hmia: Biscayne Bay
iniki.rex	hkwa: Kauai, Hawaii (Non-Op)	tpa1921.rex	etpa: Elliptical Tampa Bay
bob.rex	ny2: New York	ms2-1915.rex	ms2: New Orleans
hugo.rex	hchs: Charleston Harbor	ms2-1909.rex	ms2: New Orleans
florence.rex	ms2: New Orleans	ms2_1909.rex	ms2: New Orleans
glorany.rex	ny2: New York	gls1900.rex	egl2: Ell Galveston Bay (2002)
Alicia.rex	egl2: Ell Galveston Bay (2002)	allen.rex	br2: Laguna Madre

Figure 45 – SLOSH Historical Rex DataFiles on Install CD

The basic instruction to display historical rex file data is described below:

- 1) In the SLOSH Display Program launch the *Animation Setup* interface:
 - Menu Bar: **Animate -> Animate .rex file**
- 2) Select the desired storm and basin
 - Bullet 1: **<from provided list>**
- 3) Animate selected storm
 - Bullet 6: **Start**

The features and functionality of the *Animation Setup* interface shown in figure 46:

1. Storm & Basin Selection: Selectable *Animation Filename* and *Basin Name* list box of predefined, historical, .rex data files. Listed files are stored on the local drive at:

C:\slosh.pkg\data\rexfiles\years\<year>. File naming conventions vary, however most common used is in the format described by example below:

- *c:\slosh.pkg\data\rexfiles\years\1999\geor_ms2.rex*
 - geor = Storm Name or partial name
 - ms2 = basin (New Orleans)
 - rex = rex file identification tag
- *c:\slosh.pkg\data\rexfiles\years\1999\K27_ms2.rex*
 - K = Storm name noted by first letter
 - 27 = Forecast advisory #
 - ms2 = basin (New Orleans)

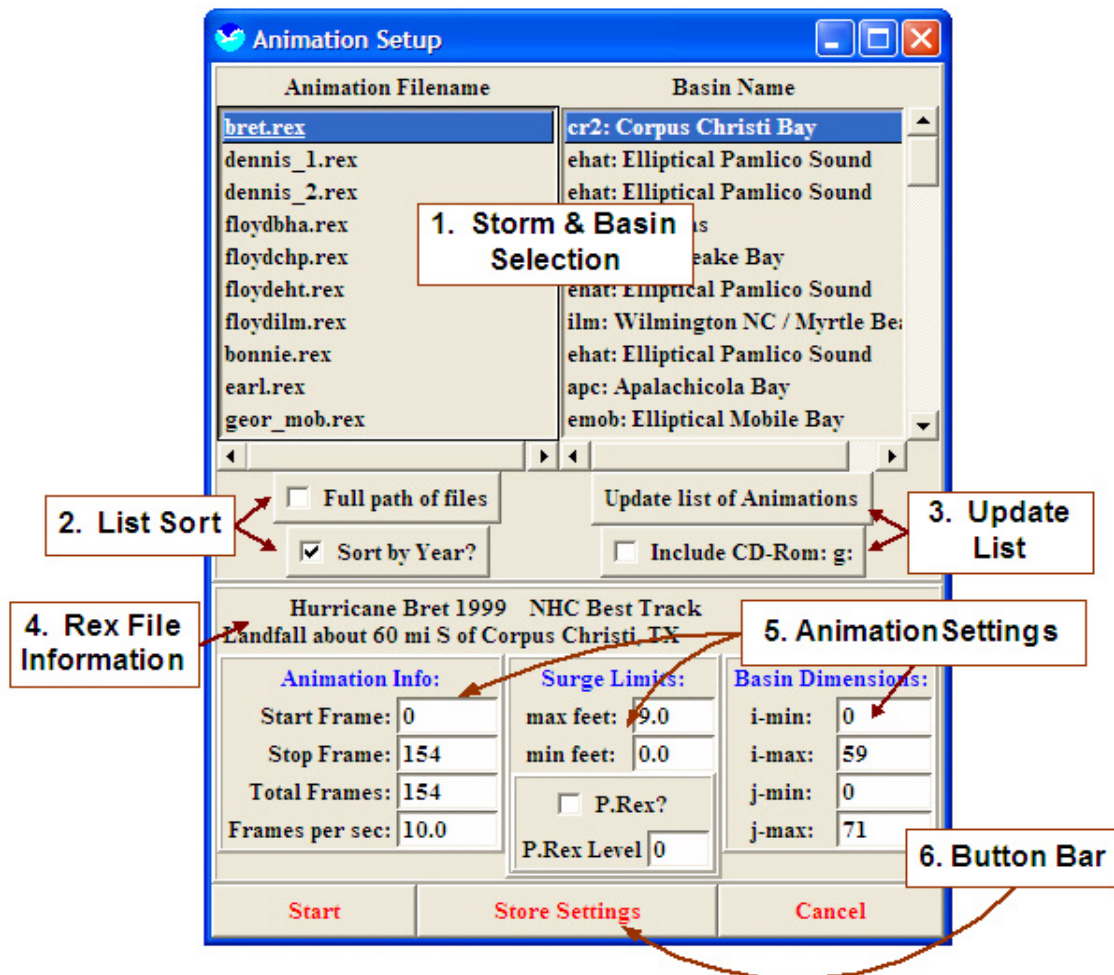


Figure 46 – SLOSH Display Program’s Animation Setup Interface

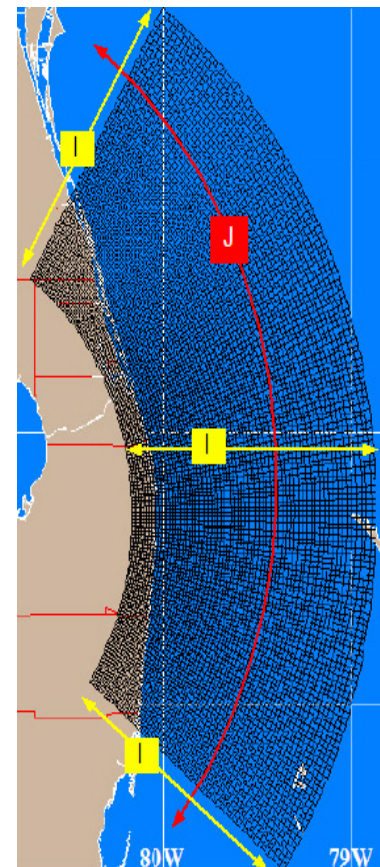
2. Sort List: To assist user in finding the desired storm selection two sorting features are provided: 1) Full path of files – adding the .rex full directory path information to *Animation File* list box data, default directory is *C:\slosh.pkg\data\rexfiles\years\<year>*, and 2) Sort by year – which orders *Animation File* list box data by earliest year of storm data.

3. Update List: Update list button allows users to retrieve imported .rex file storm data placed in the *C:\slosh.pkg\data\rexfiles\years* directory a CD-Rom search for .rex file will also be made if the *Include CD-Rom* option is checked. Updated results will be included in the *Animation File* and *Basin Name* list boxes.

4. Rex File Information: Information label describing the selected storm as reflected in the Rex file.

5. Animation Settings: Rex files have default animation settings, set by the person who created the original file. These parameters were selected to best represent the selected storm data, however users can temporarily adjust these values or permanently for the local copy by selecting *Store* after making any changes.

- Animation Info:
 - Start Frame: Begin loop at frame #
 - Stop Frame: End loop at frame #
 - Total Frames: Total number of individual frames that comprise rex image loop
 - Frames per second: Number of frames displayed in loop per second
- Surge Limits:
 - max feet: Maximum storm surge values on SLOSH map
 - min feet: Minimum storm surge values to display on SLOSH map
 - P.Rex & P.Rex Level: Future probabilistic storm surge settings to be fully defined at a later date.
- Basin Dimensions:
 - SLOSH grid dimensions closely resemble lat/lon, however due to the curvature SLOSH coordinates use a I/J representation as shown in figure 47.



6. Button Bar: The button bar consists of 3 buttons; 1) *Start* - begin animation of selected storm on SLOSH display, 2) *Store Settings* - will save, overwrite, original rex file animations parameters of selected Rex file.

Figure 47 – SLOSH I/J Coordinates

Consider the following when displaying an animated .rex file:

- The storm surge will automatically be displayed on the animation. To view the animation without storm surge turn off the *Surge* option under *Animate* on the menu bar.

- Wind directions can also be displayed along with the animation. The winds can be activated by turning on the *Winds option* under *Animate* on the menu bar. These winds are the winds that the SLOSH model used in its computations and are approximately the ten minute average winds
- A new toolbar will appear:
 - *Stop Sign / Exit Animation Button* – will quit the animation function and restore the slosh display.
 - *Pause* button – will pause the animation.
 - *Time History* – function for users to interrogate grid cell specific storm surge data over time – see section 5.8.1.
 - *Forward To End* – advances the animation to the last frame when paused.
 - *Forward 1 Frame* – advances the animation one frame when paused.
 - *Rewind 1 Frame* – backs up the animation by one frame
 - *Rewind To Beginning* – backs the animation to the starting frame

5.5 Importing a Rex File

It is sometimes necessary to import actual National Hurricane Center (NHC) Rex based SLOSH forecasts or Rex files shared between storm surge researchers. The following steps can be used to import a Rex file into the SLOSH Display Program:

- 1) Launch the “SLOSH Display Program” application
- 2) Launch “Import REX” GUI:
 - ▶ Menu Bar: **Animate -> Import .rex file**
 - ▶ Click “OK” to message:
 - *“This allows you to import a .rex file from an arbitrary place, copy it to c:/slosh.pkg/data/rexfiles and let the SLOSH display program know you did so. First step...select the files to import. Continue?”*
- 3) In the *Import REX Files* interface:
 - ▶ **Navigate to desired directory**
 - ▶ **Select desired REX file**
 - ▶ **Select Done**
 - ▶ Click “OK” to several message boxes displayed:
- 4) Done

5.6 Generating PCX Based Animated GIF Images

- 1) Launch the “SLOSH Display Program” application
- 2) Launch “Import REX” GUI:
 - ▶ Menu Bar: **Animate -> Import .rex file**
 - ▶ Click “OK” to message:

- “This allows you to import a .rex file from an arbitrary place, copy it to c:/slosh.pkg/data/rexfiles and let the SLOSH display program know you did so. First step...select the files to import. Continue?”

3) In the *Import REX Files* interface:

- ▶ **Navigate to desired directory**
- ▶ **Select “all” desired REX files**
- ▶ **Select Done**

4) Animate REX files:

- ▶ Menu Bar: **Animate -> Animate .rex file -> select file**
- ▶ Select: **Just Import -> Start**

Note: Allow loop to run through once before moving on to step 5

5) Remove Wind & Basin grids:

- ▶ Menu Bar: **Animate -> No Wind**
- ▶ Menu Bar: **Display -> SLOSH Grid**

6) Resize map, zoom as needed, for best centering:

- ▶ Vertical Menu Bar: **<Magnify Glass Button>**

7) Place map keys in appropriate locations as desired:

- ▶ **Drag & Drop with left mouse**

8) Place appropriate “Probe Flags” to determine water level for desired locations. *Ensure flags are over water rather than over land:*

- ▶ **Left mouse selection, flag on left vertical button bar.**
- ▶ **On the map, move curser over desired area, single click left mouse to place**
- ▶ **Repeat flag placement as desired**

9) Save to SHP:

- ▶ Menu Bar: **File -> Save to SHP**

10) In “Save As” GUI select folder to save in:

At TPC use the following steps:

- ▶ Change Directory: **w:**
- ▶ Select Drive Folder: **G-LOCKET**
- ▶ Select Storm Folder: **<Katrina2005>**
- ▶ Name File:

<run>_<basin>_<added water in feet>_<RMAX>_<SLOSH Version>_<Date>.rex
EG: final_ms2_2_4-rmw_3-5slosh_01_24_2006.rex

- ▶ Select: **Done**

11) In “Ship Query” GUI displayed, do nothing:

- ▶ Select: **Done**

12) Animate:

- ▶ Menu Bar: **Animate -> Animate to PCX**
- ▶ In GUI Select: **OK**
At TPC use the following steps
Keep Default Directory
- ▶ “Deleting Old” message box Select: **OK**
Long Processing

13) Load “Anamagic Software”:

- ▶ At TPC: **Desktop Icon**

14) Open New Session:

- ▶ Menu Bar: **File -> Open**

15) Change file type to PCX:

- ▶ Bottom Dropdown Menu Select: **PCX**

16) Select PCX Files:

Steps for < 191 PCX Files

- ▶ Hold “Shift Key”, Left mouse select first image
- ▶ Select the last PCX file
Will highlight the first through the last
- ▶ Select: **OK**

Steps for > 191 PCX Files

- ▶ **Hold “Shift Key”, Left mouse select first image**
- ▶ **Select the 191th PCX file**
The first 191 files will be highlighted
- ▶ Select: **OK**
- ▶ Menu Bar: **File -> Append Frames...**
- ▶ **Hold “Shift Key”, Left mouse select first image**
- ▶ **Select the 191th PCX file, or last**
- ▶ Select: **OK**
- ▶ Repeat “Append Frames...” steps until all desired PCX files have been loaded

17) Set Animation Frame Rate:

- ▶ Menu Bar: **Animation -> Frame Rate**
- ▶ Set: **100** <milliseconds>

18) Test Animation, Visually inspect for bad or undesired frames:

- ▶ Vertical Button Bar Select: **Play**

19) Delete bad or undesired frames as needed:

20) Repeat Steps 18 & 19 until loop/animation is desirable:

21) Set Frame Rate Animation:

- ▶ Menu Bar: **Animation -> Frame Rate**

- ▶ Set “First Frame” to: **15**
- ▶ Set “Last Frame” to: **25**

22) Watch Animation again for accuracy, make changes as needed

23) Save File:

- ▶ Menu Bar: **File -> Save Optimized**
- ▶ **Select Desired Directory**
- ▶ Name File: **<use same name in step 10>**
- ▶ Select: **Save**

24) Done

5.7 Astronomical Tide Prediction

Tides are generated by the gravitational attraction between both the earth and the moon and the earth and the sun. Tides vary significantly in amplitude, maxima/day, and minima/day along the U.S. coastline depending on latitude, geographic location, and bathymetry. Astronomical tide computations use mathematical expressions containing unique constituents for each location. Tidal constituents are determined by analyzing tide gauge data from anywhere between several months to the ideal period of 19 years (or a tidal epoch). Tide elevations are evaluated assuming the bathymetry of the location remains unchanged since the constituents were derived.

Tides play an important role in determining the total water level experienced during a hurricane. SLOSH currently computes the wind-induced storm surge above a constant tide level. Generally, SLOSH MOMs and MEOWs were formed using high-tide water conditions. The *Astronomical Tide Prediction or Tide Display* program allows forecasters and emergency managers to view the range of predicted tides for a given period.

The Computations used in the *Tide Display Program* were created by Art Pore and modified by Kurt W. Hess. Tide computations should be checked periodically for accuracy against the NOS tide tables, especially for critical situations. To further understand how tides interact with storm surge heights reference the extratropical storm surge website at: <http://nws.noaa.gov/mdl/etsurge>. This site is an extremely useful source of information about the predicted total water level at a given station. Figure 49 is a snapshot of the *Tide Display* interface launched from the *SLOSH Display* menu bar under Tide -> Tide Display, the features and functionality break down as follows:

Woods Hole, MA		
date	hr	height (Feet)
03/01/2006	00	-0.24
03/01/2006	01	-0.63
03/01/2006	02	-1.13
< snip >		
Stamford, CN		
date	hr	height (Feet)
03/01/2006	00	4.99
03/01/2006	01	4.14
03/01/2006	02	2.21
< snip >		
Bar Harbor, ME		
date	hr	height (Feet)
03/01/2006	00	6.36
03/01/2006	01	3.99
03/01/2006	02	0.54
< snip >		

Figure 48 - .txt Tide Data

Menu Bar Option: **File**:

Save (.ps): Save Tide Display as PostScript image file.

Save (.pcx): Save Tide Display as a standard .pcx image file

Save (.txt): Save hourly tide data of selected stations to an ASCII text file – see figure 48

Print: Print Tide Display Image – Deactivated.

Exit: Exit Tide Display Program

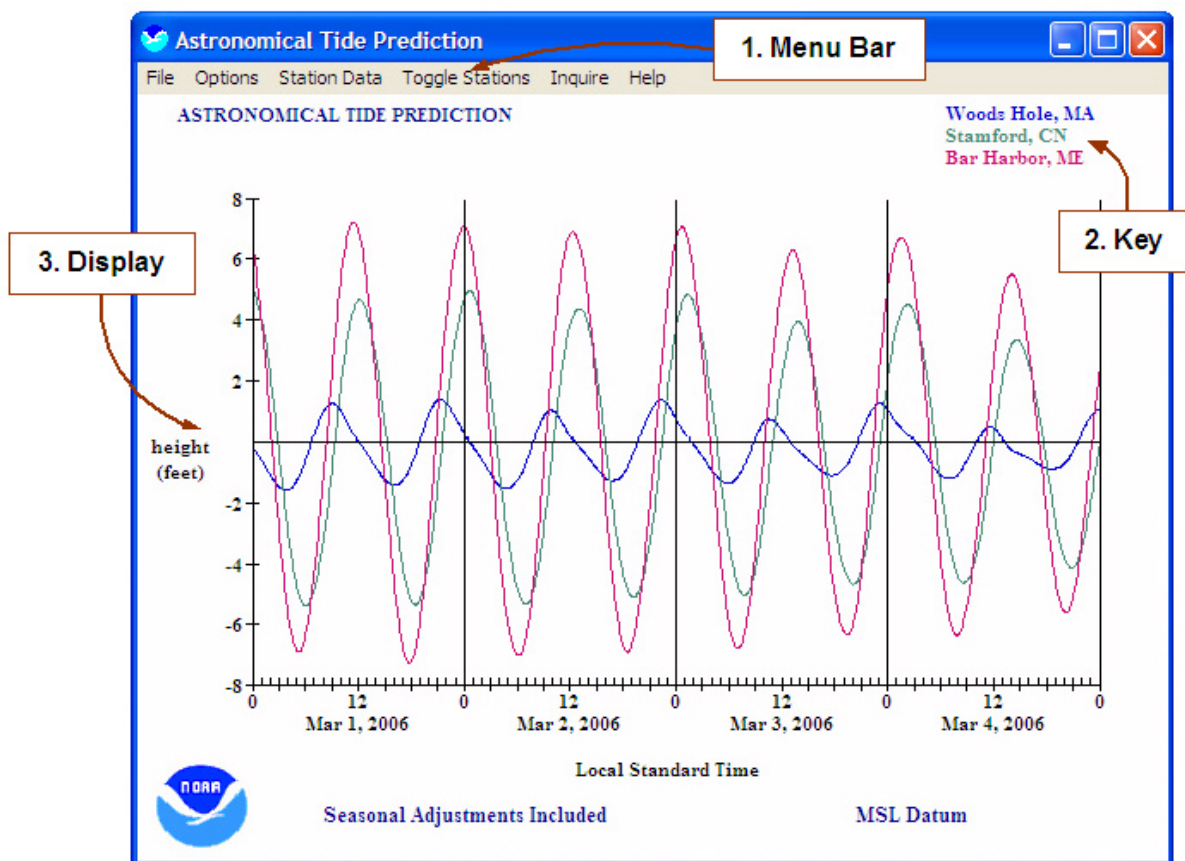


Figure 49 – Astronomical Tide Prediction or Tide Display Program

Menu Bar Option: **Options**:

View Data: Graphical User Interface display of hourly tide data of selected stations. Similar to Save (.txt) function.

Units: Selections to toggle displayed data between feet and meters.

Grid: Selection to toggle “whole number” x, y grid on data display. Default is inactive, not shown in figure 50.

Change Background Color: Launches a color pallet allowing user to change the Tide Display background from default white to any selected color.

Change Station Colors: Color pallet to change selected side station colors.

Restore Default Colors: Restores Default colors for both the display background and selected tide stations.

Modify Graph Parameters: Allows users to adjust the X-Axis, Y-Axis, and Y Increment parameters of the tide display – see figure 50.

Restore Parameters: Function to reset defaults of the tide display altered by *Modify Graph Parameters* function.

Restore Label Positions: Label positions can be moved with left mouse button, this option places this key data back to the default placement as shown.

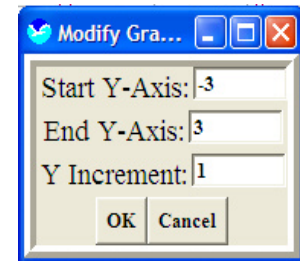


Figure 50 - Modify Graph GUI

Woods Hole, MA			Stamford, CN			Bar Harbor, ME		
date	hour	height (Feet)	date	hour	height (Feet)	date	hour	height (Feet)
03/01/2006	00	-0.24	03/01/2006	00	4.99	03/01/2006	00	6.36
03/01/2006	01	-0.63	03/01/2006	01	4.14	03/01/2006	01	3.99
03/01/2006	02	-1.13	03/01/2006	02	2.21	03/01/2006	02	0.54
03/01/2006	03	-1.51	03/01/2006	03	-0.40	03/01/2006	03	-3.01
03/01/2006	04	-1.61	03/01/2006	04	-2.97	03/01/2006	04	-5.75
03/01/2006	05	-1.35	03/01/2006	05	-4.80	03/01/2006	05	-7.06
03/01/2006	06	-0.69	03/01/2006	06	-5.53	03/01/2006	06	-6.66
03/01/2006	07	0.22	03/01/2006	07	-5.05	03/01/2006	07	-4.54
03/01/2006	08	1.03	03/01/2006	08	-3.36	03/01/2006	08	-1.16
03/01/2006	09	1.35	03/01/2006	09	-0.77	03/01/2006	09	2.54
03/01/2006	10	1.05	03/01/2006	10	1.95	03/01/2006	10	5.56
03/01/2006	11	0.46	03/01/2006	11	3.96	03/01/2006	11	7.23
03/01/2006	12	0.01	03/01/2006	12	4.80	03/01/2006	12	7.20
03/01/2006	13	-0.35	03/01/2006	13	4.41	03/01/2006	13	5.41
03/01/2006	14	-0.82	03/01/2006	14	2.89	03/01/2006	14	2.24
03/01/2006	15	-1.27	03/01/2006	15	0.50	03/01/2006	15	-1.49
03/01/2006	16	-1.46	03/01/2006	16	-2.13	03/01/2006	16	-4.80
03/01/2006	17	-1.31	03/01/2006	17	-4.26	03/01/2006	17	-6.92
03/01/2006	18	-0.78	03/01/2006	18	-5.38	03/01/2006	18	-7.41
03/01/2006	19	0.06	03/01/2006	19	-5.32	03/01/2006	19	-6.06

Figure 51 – Tide Data User Interface, View Data

Menu Bar Option: **Station Data:**

Change Current Stations: Provides a list of 179 selectable tide stations to plot on the *Tide Display* (see table below) and a max of three stations can be selected at one time see figure 51 for snapshot of interface. All or some of the stations can be changed by first double clicking on the desired station and then clicking ok. If duplicate stations are chosen, the graphs will be drawn overlapping.

North East:		
1 Eastport, ME	12 Providence, RI	23 Willets Point, NY
2 Bar Hbr, Frenchman Bay, ME	13 New Bedford, MA	24 New York (Battery), NY
3 Bar Harbor, ME	14 Newport, RI	25 Albany, NY
4 Portland, ME	15 Block Island, RI	26 Bayonne Br., Staten Is. NY
5 Rockland, ME	16 New London, CN	27 Bay Shore, L.I., NY
6 Portsmouth, NH	17 Bridgeport, CN	28 Spuyten Cr., Hudson R., NY
7 Boston, MA	18 Stamford, CN	29 Sandy Hook, NJ
8 Cape Cod, Sandwich, Ma	19 Old Saybrook Point, CN	30 Atlantic City, NJ
9 Cape Cod, Buzzards Bay, MA	20 Montauk Point, NY	31 Cape May, NJ
10 Nantucket, MA	21 Montauk, Ft. Pond Bay, NY	32 Trenton, Del R., NJ
11 Woods Hole, MA	22 Port Jefferson, NY	33 Tuckahoe River, NJ
Mid-Atlantic:		
1 Philadelphia, PA	10 Chesapeake Beach, MD	19 Chincoteague Channel, VA
2 Philadelphia, Del R., PA	11 Havre De Grace, Susq R, MD	20 Colonial Beach, VA
3 Reedy Point, DE	12 Tolchester Beach, MD	21 Clouchester Pt, VA
4 Breakwater Harbor, DE	13 Washington, DC	22 Kiptopeke Beach, VA
5 Indian River Inlet, DE	14 Solomons Is, MD	23 Lewisetta, Potomac R, VA
6 Ocean City, MD	15 Old Point Comfort, VA	24 Wachapreague, VA
7 Baltimore, MD	16 Hampton Roads, VA	25 Wallops Is., VA
8 Annapolis, MD	17 Chesapke Bay Br Tunnel VA	26 Windmill Pt, Ches Bay, VA
9 Cambridge, MD	18 Portsmouth, VA	27 Virginia Beach, VA
Southeast:		
1 Duck Pier, NC	10 Myrtle Bch (Springmd P), SC	19 Daytona Beach, (Ocean) FL
2 Avon, NC	11 Myrtle Beach, SC	20 Daytona Bch, (Sunglo P.) FL
3 Cape Hatteras (Pier), NC	12 Charleston, SC	21 Canaveral Harbor Ent, FL
4 Morehead City, NC	13 Cooper River, SC	22 Lake Worth Pier, FL
5 Wilmington, NC	14 Savannah, GA	23 Haulover Pier, FL
6 Wrightsville Beach, NC	15 Savannah R. Entrance, GA	24 Miami, FL
7 Holden Beach, NC	16 Kings Bay, Cumberland GA	25 Vaca Key, Florida Bay, FL
8 Beaufort, Duke, NC	17 Fernandina, FL	26 Key West, FL
9 Ocracoke, NC	18 Mayport, FL	
Gulf of Mexico:		
1 Naples, FL	23 Ozello, St Martines R. FL	45 Eugene Island, LA
2 Punta Rasa, FL	24 Panama City Beach, FL	46 Weeks Bay, LA
3 Anna Maria, FL	25 Red Bay Pt. St Johns R. FL	47 Calcasieu Pass, LA
4 St. Petersburg, FL	26 Shell Is., Crystal R. FL	48 Terrebonne Bay, LA
5 Indian Bay, FL	27 Shell Point, FL	49 Cypremort Pt, LA
6 Cedar Key, FL	28 St. Augustine Beach, FL	50 Southwest Pass (Delta) LA
7 St. Marks R. Entrance, FL	29 Port Canaveral, FL	51 Venice, Grand Pass, LA
8 Apalachicola, FL	30 Tuckers Is, Homosassa, FL	52 Sabine Pass, TX
9 Aligator Bayou, FL	31 Turkey Point, FL	53 Gilchrist, TX
10 Pensacola, FL	32 Twin R Marina, Crystl R FL	54 Galveston, TX
11 Chassahowitzka R., FL	33 Virginia Key, Bear Cut, FL	55 Galveston, TX Pleasure P.
12 Clearwater Beach, FL	34 Mobile, AL	56 Freeport, TX
13 Dixie Bay, Salt R., FL	35 Dauphin IS., AL	57 Pt O'Connor, Matagorda TX

14 Fort Myers, FL 15 Halls River, FL 16 Chassahowitzka Bay, FL 17 Key Colony Beach, FL 18 Kings Bay, Crystal R. FL 19 Mangrove Pt, Crystal R. FL 20 Homosassa Bay, FL 21 Navarre Beach, FL 22 Ozello N., Crystal Bay FL	36 Bay St. Louis, MS 37 Waveland, Bay St. Louis MS 38 Gulfport Harbor, MS 39 Waveland, MS sound, MS 40 South pass, LA 41 Grand Is, East Pt, LA 42 Bayou Rigaud, LA 43 Humble Oil Platform, LA 44 Timbalier Island, LA	58 Port Aransas, TX 59 Port Isabel, TX 60 Padre Island, TX 61 Aransas Channel, TX 62 Corpus Christi, TX 63 Nueces Bay, TX 64 Lynchburg, TX 65 Pt Lavaca, Matagorda TX 66 Rockport, Aransas Bay TX
Caribbean:		
1 San Juan, PR 2 Boca De Cangrejos, PR 3 Mangueyes Island, PR	4 Mangueyes Island, PR 5 Punta Tuna, PR 6 Brenner Bay, St Thomas VI	7 Charolette Amali, VI 8 St Croix, Lime Tree, VI
Pacific:		
1 Hilo Bay, HI 2 Honolulu, Oahu Is, HI 3 Kahului Harbor, HI 4 Kahului, Maui Is, HI 5 Kawaihae, Hawaii Is, HI 6 Kaneohe Bay, Oahu HI 7 Kaneohe Bay, Oahu HI	8 Kaneohe Bay, Oahu HI 9 Nawiliwili, HI 10 Sand Is, Midway Islands 11 Guam, Marianas 12 Johnston Atoll, Pac Ocean 13 Wake Is, Pacific Ocean 14 Kwajalein Atoll, Marshl Is	15 Malakal Harbor, palau Is 16 Moen Is, Truk Atoll, Pac Oce 17 Pago Pago, American Samoa 18 Ponape Hbr, Caroline Is 19 Mcmurdo Sound, Antarctic

Figure 52 – Tide Stations in SLSOH

Change Starting Date: Produces a window containing four X-axis parameters: 1) Starting Year, 2) Starting Month, 3) Starting Day, and 4) Number of Days Shown. Each of these parameters may be modified by entering the desired number (within the boundaries given in parenthesis) and then clicking on OK. The Change Starting Date window contains an error check for leap years.

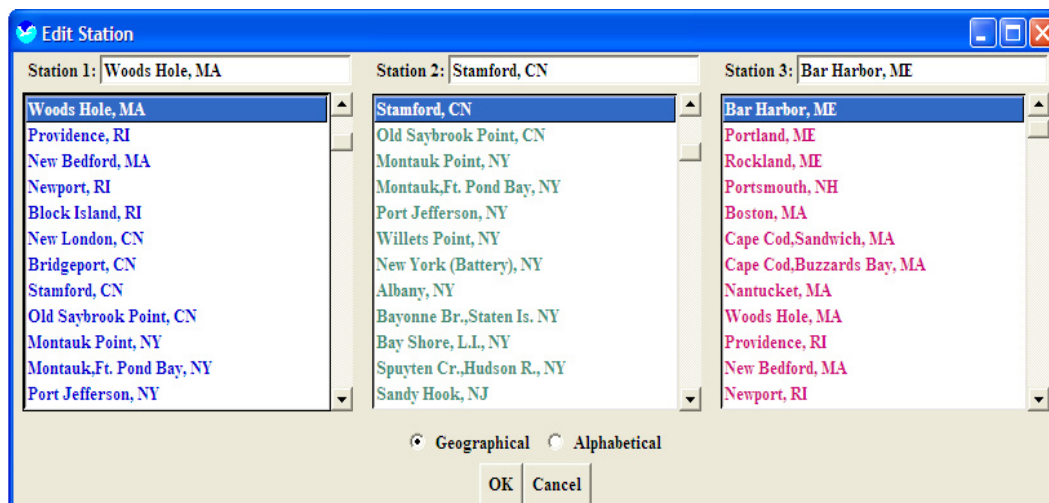


Figure 53 – Change Current Station / Edit Station Tide Display Interface

Adjust Reference Levels: The default reference level (or 0) for all tide data is set at Mean Sea Level (MSL). This option allows users to set a reference level above or below MSL. To restore back to ML users can select the *Restore MSL Datum* option.

Include Seasonal Adjustments (SA and SSA): The Tide Display Program automatically includes seasonal adjustments (SA & SSA) in its tide constituent data. Normal tide values always include these constants. If unselected this function allows users to view tide data adjustments.

Menu Bar Option: **Toggle Stations:**

Station 1,2,3: Toggle display of selected stations 1, 2, and 3.

Menu Bar Option: **Inquire:**

Inquire: Allows users to view station x and y data by hour . Launching this application produces a red (or white pending background color) vertical line in the middle of the tide graph, bullet 3. The graph can be traced using the [Left] and [Right] arrow keys or the mouse. Y-coordinate (height) points are displayed next to the station name labels, label 2. Updated date and time labels are also displayed in the top center, label 1.

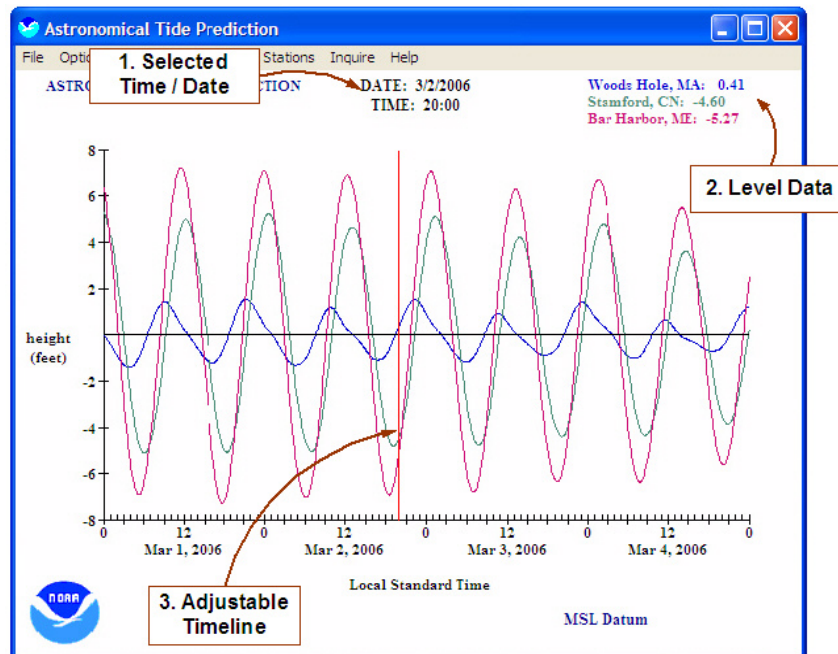


Figure 54 - Tide Display Program Highlighting Inquire Functionality

Menu Bar Option: **Help:**

Help: The *Tide Display Program* help function launches an HTML based interface providing help on select SLOSH topics.

Credits: SLOSH Version, Launch date, and disclaimer statements.

5.8 Storm Surge Time History

The *Time History* function allows users to interrogate grid cell specific storm surge data over time – see figure 55. The following step-by-step instructions describe the basic use.

- 1) Launch the “SLOSH Display Program” application.
- 2) Animate a Rex file see section 5.4.
- 3) Select the *Time History* button, last button on the vertical button bar of the SLOSH Display Program.
- 4) Place the mouse over desired area on map display and select the left mouse button and repeat as needed
- 5) Modify data parameters as desired.

5.8.1 Storm Surge Time History Features and Functionality

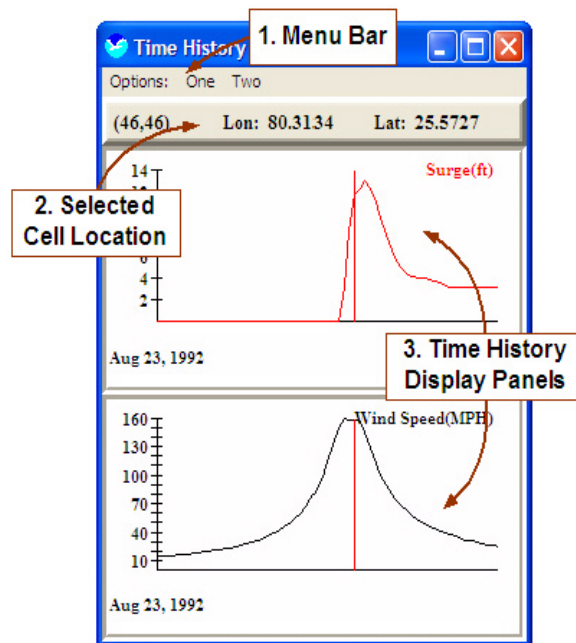


Figure 55 – Time History Interface

This section describes a detailed look at the features and functionality provided in the *Time History* program – see figure 55.

Menu Bar Option (Label 1): Options:

Close: Close the Time History interface

Save to .txt: Save Time History data in a tab delimited ASCII text file format. Data saved to file when using this function is: 1) surge, 2) storm speed, 3) Storm Delta P, 4) Storm Rmax, and 5) Distance - See figure 56 for example format.

```

Basin=eht Elliptical Pamlico Sound
RexFile=c:/slosh.pkg/data/rexfiles/years/1999/floydeht.rex
Grid Reference (152,45) Lat= 34.5351 Lon= 77.0956
Date      Time      Surge WindSpeed  Center_Lat  Center_Lon  Del_P      R_Max
09/15/1999 18:01:00  2.3    26.57    30.608034  79.099098  62.983330  39.988998
09/15/1999 18:15:00  2.3    26.84    30.664261  79.092804  62.866638  39.911987
09/15/1999 18:30:00  2.3    27.13    30.724510  79.086067  62.741611  39.829475
09/15/1999 18:45:00  2.3    27.43    30.784760  79.079308  62.616585  39.746964
09/15/1999 19:00:00  2.3    27.55    30.845110  79.072197  62.491665  39.664333
<...snip...>

```

Figure 56 – ASCII Text File Format of Time History Program Data

Save to .ps: Save Time History display to standard postScript image format.

Save to .pcx: Saves Time History display as standard .pcx graphic image.

Number of graphs: Cascading menu allowing users to select up to five x and y panels which also expands the menu bar by the number of selected graphs/panels. By default two graphs are selected. It may be necessary to increase the computers display resolution to view all graphs depending on the existing display resolution setting. You will know if this is necessary because the Time History interface will run off the screen.

Menu Bar Option (Label 1): One ... Five: Provides on/off toggle of the described parameters below. Note: By default the menu bar will have two “graph display” options which is expandable as described in the *Number of graphs* function above. Whether one or five, the functionality presented under the menu bar is the same as described below.

Figure 57 – Modify Time History Graph Parameters Interface

Surge: Display time graph of storm surge values for selected cell.

Speed: Display time graph of the one minute average wind speed (sustained/mpg) at the selected location.

Delta P: Change in pressure (mb) at selected cell location.

Storm Rmax: Static Rmax value (STM file) as provided by forecaster creating model run.

Distance: Distance storm eye is from the selected point in miles.

Modify Parameters: Provides users an interface (figure 57) to modify the parameters of the selected panel/cell as described below:

- X-Start – Define the value to which the x-axis begins
- X-End – Define the ending value of the x-axis
- X-Increment – Define the increment between values for the x-axis
- X-Label – Label the x-axis as desired, no label by default
- Y-Start – Define the value to which the y-axis begins
- Y-End – Define the ending value of the y-axis
- Y-Increment – Define the increment between values for the y-axis
- Y-Label – Label the y-axis as described, no label by default.

Selected Cell Location: Highlights the location of the selected cell by SLOSH grid and Lat/Lon locations.

Time History Display Panels: Highlighting two of five possible display panels for time height data. X-axis is time in hours and the Y-axis are generic units supporting mixed type depending on overlays.

Appendix A

Acronyms / Definitions

- **SLOSH:**
Sea Lake and Overland Surges from Hurricanes
- **SLOSH Model:**
The model generating hurricane storm surge given a **SLOSH Basin** and Hurricane Track, also generating an **Envelope** of high water and **Rex File**
- **SLOSH grid:**
The **SLOSH** model uses a polar, elliptical, or hyperbolic grid for its computations. Cartesian grids are not used.
- **SLOSH Basin:**
A geographic region as defined by the **SLOSH grid** with known values for topography, and bathymetry in addition to grid properties and barriers, on which the SLOSH model is run.
- **Hurricane Track:**
A set of parameters used to approximate the hurricane which includes: 1) **Pressure, Radius of Maximum Winds (RMAX)**, Location, Forward Speed, and Forward Direction.
- **Envelope:**
Maximum water level at any given point at every grid cell in the **SLOSH basin**.
- **Rex file:**
The water level height at every grid cell, in a defined **SLOSH Basin**, at a specific interval of time. The SLOSH display program displays Rex files as time-lapsed animated “movies”.
- **Maximum Envelope of Water (MEOW):**
Composite of maximum storm surge heights at each grid cell using hypothetical hurricane runs with the same: 1) **category**, 2) forward speed, 3) landfall direction, 4) initial tide level, and 5) radius of maximum wind. Composite achieved by reviewing parallel tracks that make landfall at different locations.
- **Category:** The category of hurricane, based on the Saffir-Simpson Scale:

Type	Maximum Wind Speed		Pressure mb
	Knots	MPH	
Depression	< 34	< 39	---
Tropical Storm	34-63	39-73	---
Category 1	64-82	74-95	> 980
Category 2	83-95	96-110	965-980
Category 3	96-112	111-130	945-965
Category 4	113-134	131-155	920-945
Category 5	> 134	> 155	< 920

Figure 58 - Saffir-Simpson Scale

- **Maximum of MEOWs (MOM):**
Composite of the maximum storm surge height for all hurricanes of a given category. Disregards forward speed, landfall direction, land fall location. Typically only five MOMs per basin. For example, one per storm category; may be doubled with more than one tide level.
- **Evaluation Branch (EB):**
MDL Branch responsible for developing and maintaining SLOSH Software models.
- **Meteorological Development Laboratory (MDL):**
NWS Division under the OST responsible for developing techniques (programs) to aid WFOs and NCEP National Centers.
- **Mean Lower Low Water (MLLW)**
The average height of the lower low waters over a 19-year period. For shorter periods of observation, corrections are applied to eliminate known variations and reduce the result to the equivalent of a mean 19-year value.
(<http://www.ecy.wa.gov/>)
- **Mean Seal Level (MSL)**
The average height of the surface of the sea for all stages of the tide over a 19-year period, usually determined from hourly height readings
(<http://www.ecy.wa.gov/>)
- **National Hurricane Center (NHC) / Tropical Prediction Center (TPC):**
NWS NCEP agency charged with tropical storm prediction.
<http://www.nhc.noaa.gov>
- **National Weather Service (NWS):**
One of 5 NOAA line offices. NWS is charged with weather prediction and warning to protect life and property. <http://nws.noaa.gov>
- **National Oceanic And Atmospheric Administration (NOAA):**
DOC agency focused on Ocean and Atmospheric concerns
- **Department of Commerce:**
Government agency focused on US Commerce. <http://www.doc.gov>
- **Continental Shelf:**
The extended perimeter of each continent.

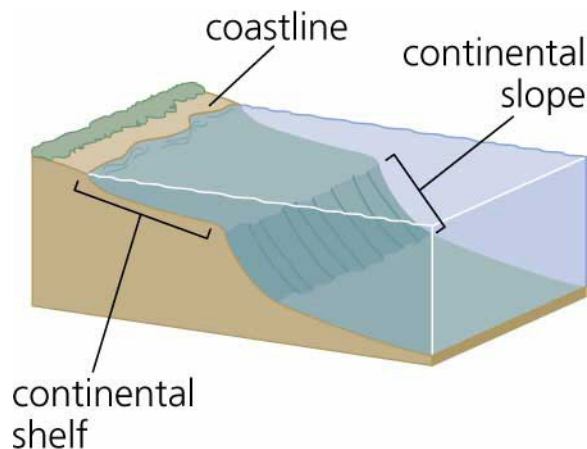


Figure 59 – Continental Shelf

- **Title Datum:** Image and definitions from NOS http://www.co-ops.nos.noaa.gov/datum_order.shtml

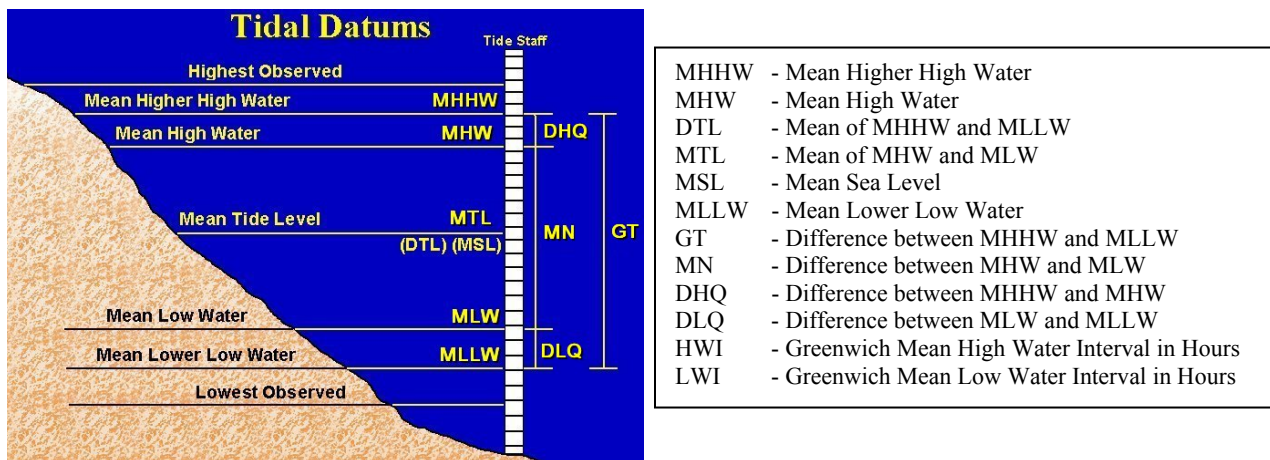


Figure 60 – Tidal Datums