Documentation of a web based source code library for WAM

Reference: MyWave- D1.1

Project N°: FP7-SPACE-2011-284455	**Work programme topic:** SPA.2011.1.5.03 – R&D to enhance future GMES applications in the Marine and Atmosphere areas
Start Date of project : 01.01-2012	**Duration**: 36 Months

WP leader: Peter Janssen **Issue: WP1 – Task 1.4**

Contributors : Arno Behrens (HZG)

MyWave version scope : version 0

Approval Date : 30 June 2013 **Approver:** Øyvind Sætra (NMI)

Dissemination level: PU

DOCUMENT

VERIFICATION AND DISTRIBUTION LIST

	Name	Work Package	Date
Checked By:	Joanna Staneva (HZG)	1	24 June 2013
Distribution			

CHANGE RECORD

Issue	Date	§	Description of Change	Author	Checked By
0.1	2013/6/20	all	First draft of document	Arno Behrens	Joanna Staneva
1.0		all	Document finalization		

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY AND ABREVIATIONS

DVCS	**D**istributed **V**ersion **C**ontrol **S**ystem
http	**h**yper**t**ext **t**ransfer **p**rotocol
MPI	**M**essage **P**assing **I**nterface
ssh	**s**ecure **sh**ell
SWAMP	**S**ea **Wa**ve **M**odeling **P**roject
WAM	**Wa**ve **M**odel

APPLICABLE AND REFERENCE DOCUMENTS

Applicable Documents

	Ref	Title	Date / Issue
DA 1	MyWave-A1	MyWave: Annex I – “Description of Work	September 2011

I INTRODUCTION

The third-generation wave model WAM (**Wa**ve **M**odel) has been used successfully for more than 20 years at numerous institutes worldwide for wave forecasting and hindcasting. In contrast to first and second generation models it solves the wave transport equation explicitly without any presumptions on the shape of the wave spectrum and represents the physics of the wave evolution for the full set of degrees of freedom of a two-dimensional wave spectrum.

Since in the meantime the source code of the old standard version WAM Cycle 4 (described in Komen et al. 1994 and Guenther et al. 1992) doesn't meet modern standards in software design anymore, a new improved source code has been developed in standard Fortran95, including MPI (**M**essage **P**assing **I**nterface) for parallelization purposes. A big advantage of the new state-of-the-art MyWave version WAM Cycle 4.5.4 is its high-grade modular composition which allows an easy replacement of individual parts of the code.

During the lifespan of MyWave all new software developments (e.g. improved source functions) will be transferred to HZG, corresponding updates inserted into the new version of the wave model and tested in the SWAMP test bed (The SWAMP Group, 1985). To make sure that all wave model developments of the MyWave project will be available for all participants, the software package is maintained in a web-based source code library which can be accessed by all registered users. For the MyWave project the free and open source **D**istributed **V**ersion **C**ontrol **S**ystem (DVCS) Git has been chosen. The Git system handles everything from small to very large projects with speed and efficiency and has important advantages compared with other modern systems. The corresponding GIT repository for WAM has been installed on the GitHub server: https://github.com/. During the lifespan of MyWave the WAM repository is a private one and will be changed to a public one afterwards. The present documentation includes an introduction into the Git system, a description of the WAM repository on the GitHub server, how to access the wave model as a contributor, to work with Git and furthermore a detailed manual for all the updates and extensions of the MyWave WAM Cycle 4.5.4 itself.

II DISTRIBUTED VERSION CONTROL SYSTEM GIT

For the MyWave project the distributed version control system Git is used because it has a lot of advantages compared with centralized systems. In those there is only one “master” repository, which every developer feeds their changes into. Every action must be synchronized with this central repository. And because it usually resides on a central server, each action has to pass through the network - leaving a developer unable to work if they happen to have no network connection. In **D**istributed **V**ersion **C**ontrol **S**ystems (DVCS), each developer has their own fully-fledged repository on the local computer. In most set-ups there’s an additional central repository on a server that’s used for sharing. However, this is not a requirement; every developer can perform all important actions in their local repository: committing changes, viewing differences between revisions, switching branches, etc.

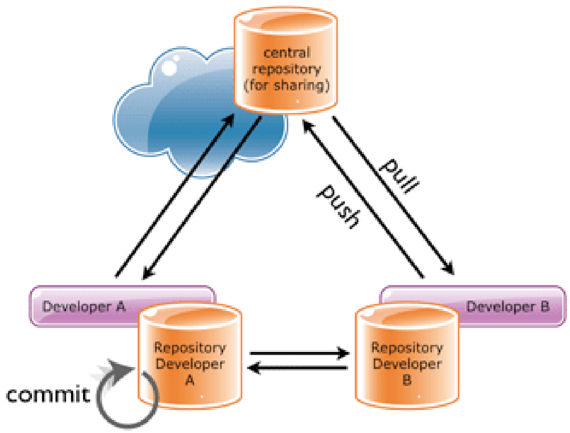

Figure 1: Layout of a distributed version control system

One of Git’s main advantages is its distributed nature. It doesn’t matter whether a complex set-up with multiple remote repositories is used or just one central server to share code (working “Subversion style”) would be available. A DVCS can be used independently of any one person’s workflow. Being able to work offline is an important advantage of DVCS for many developers. One can work without constraints, even if being not connected to the network.

Speed is another important factor, and the differences between Git and other DVCS here are evident. In almost any situation, Git is faster than other modern systems, such as Mercurial and Bazaar. One of the reasons for Git’s remarkable speed is that it was written in C. Another reason is that it was designed to work with the Linux kernel and therefore has to perform well even under huge amounts of data.

Another convenience: every local Git repository can serve as a full-fledged back-up, because it contains the project’s complete history. And considering that almost every action in Git only adds data, losing data is pretty hard to do.

The biggest advantages, however, lie in Git’s feature set: in how it deals with code and in its tools and workflows.

II.1 The WAM repository on the GitHub server

A complete set-up for one of the SWAMP cases, including the source code, makefiles for different computer systems, user input, batch jobs and output listings (to compare with) is available in a corresponding repository for the new WAM Cycle 4.5.4 on the GitHub server under the address : http://mywave.github.io/WAM/ . Figure 2 shows the start screen for the WAM repository on the GitHub server.

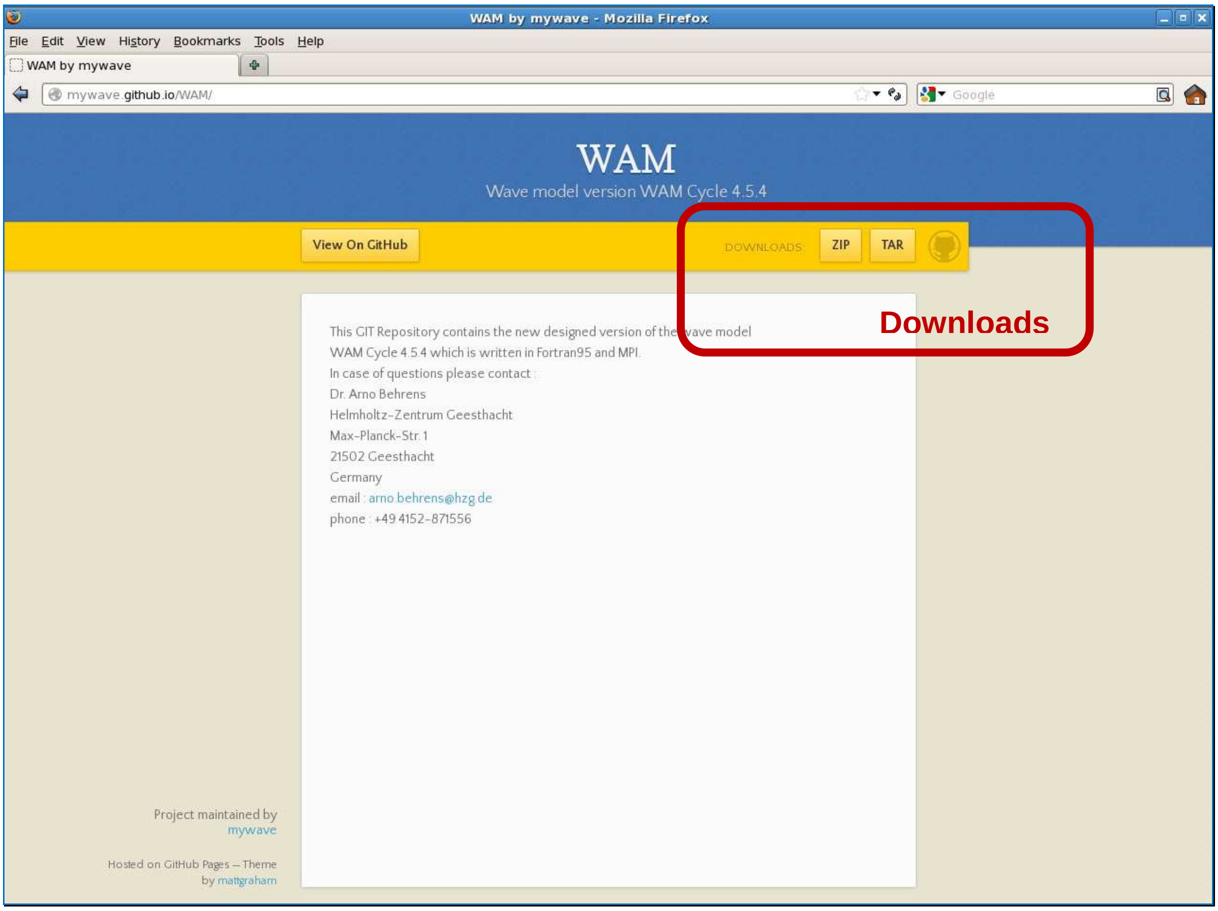

Figure 2: Start screen for the WAM repository on the GitHub server

II.2 Access to the WAM repository via http-protocol

Until the end of the MyWave project, the WAM repository is restricted to the project partners. It is not yet possible for the general user to fetch the repository - only MyWave participants who are registered can do that. Therefore all MyWave members who want to work with the wave model have to create an own account on the GitHub server with a certain arbitrary user name and password as shown in figure 3. Once performed, those will be added to the contributor list of the WAM repository by the account owner. All registered contributors have an access to the WAM repository and can download the code to their local machine for example by clicking on the download-tar-button shown in figure 2 on the top right side. If that is done the following page will appear (figure 4) and the individual contributor can log in with his/her username and password and download the complete repository as a tar-file via http-protocol to a local computer.

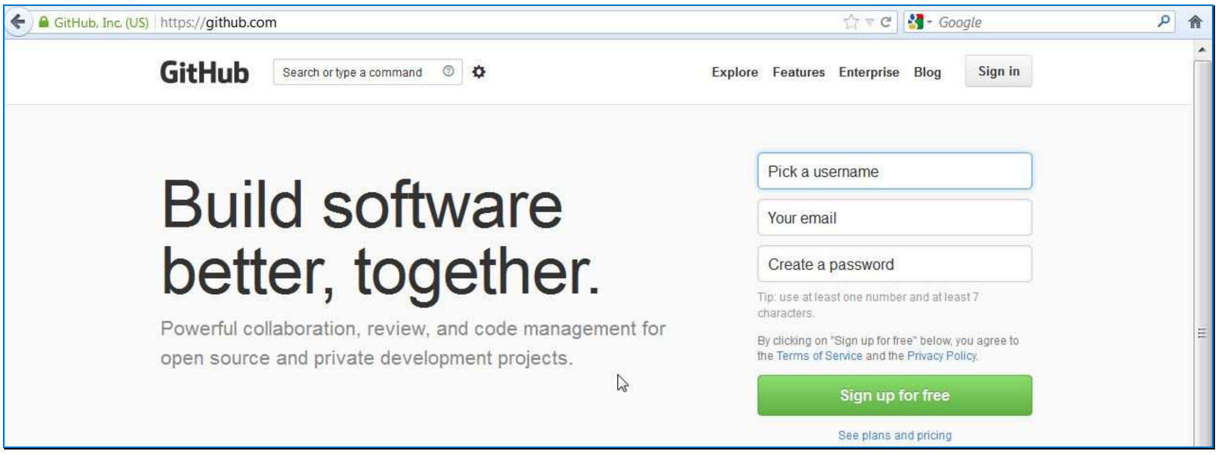

Figure 3: Registration screen for GitHub

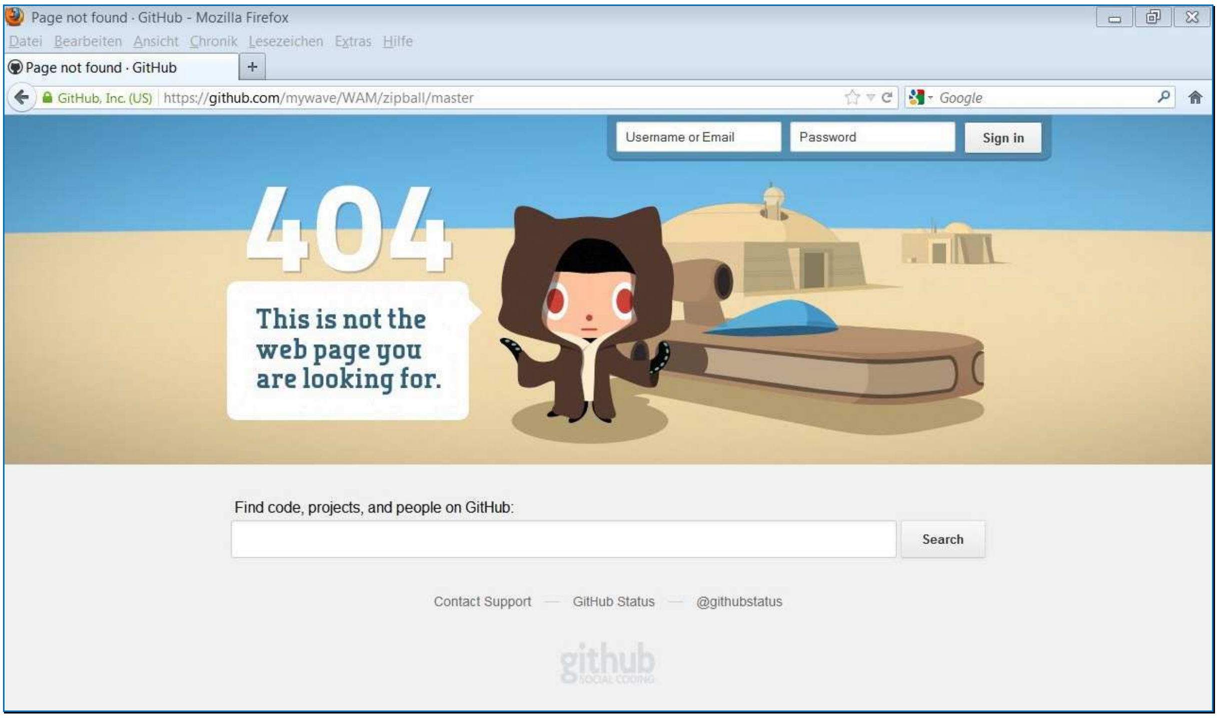

Figure 4: Login screen for registered contributors

II.3 Access to the WAM repository via ssh-protocol

Another possibility to fetch the WAM repository is an access via ssh-protocol. In that case the public ssh-key of a local remote computer is required, usually available in the home directory of the corresponding computer in the directory .ssh (file : id_rsa.pub). That key has to be added to the key list of registered computers by the account owner. In case that there is no ssh-key available on the local computer, figure 5 gives the information how to generate it. As soon as the corresponding ssh-key has been inserted into the official list of keys, it is possible to clone the complete WAM repository via ssh-protocol to a local remote computer with the following command (assumed the Git software is available on that computer) :

git clone ssh://git@github.com/mywave/WAM.git myWAM

To generate a new SSH key, enter the code below. We want the default settings so when asked to enter a file in which to save the key, just press enter.

```
$ ssh-keygen -t rsa -C "your_email@example.com"
# Creates a new ssh key, using the provided email as a label
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/you/.ssh/id_rsa): [Press enter]
```

Now you need to enter a passphrase.

Why do passphrases matter?

```
Enter passphrase (empty for no passphrase): [Type a passphrase]
Enter same passphrase again: [Type passphrase again]
```

Which should give you something like this:

```
Your identification has been saved in /c/Users/you/.ssh/id_rsa.
Your public key has been saved in /c/Users/you/.ssh/id_rsa.pub.
The key fingerprint is:
01:0f:f4:3b:ca:85:d6:17:a1:7d:f0:68:9d:f0:a2:db your_email@example.com
```

Figure 5: Instruction how to generate a public ssh-key

II.4 Working with Git

An overview of the contents of the WAM repository in mywave/WAM on the GitHub server is given in figure 6. This webpage will arise by pressing the button 'View On GitHub' (figure 2) and after signing in with a valid username and password on the intermediate page (figure 4). It shows the working environment for the WAM repository directly on the server together with the list of the available directories which includes the WAM Cycle 4.5.4 source code in 'src', makefiles for different computer systems (IBM, NEC and SUN) to generate the binaries, the constant user input files in 'const', example batch jobs for a sun cluster in 'jobs', output listings to compare with in 'dayfiles' to make sure that a remote implementation on a local computer system has been done successfully. The model set-up included in the WAM repository has been prepared for one of the SWAMP cases which will be the test bed for MyWave. Waves are generated in a rectangular basin driven by a constant wind of 18.5 m/s to the north as shown in figure 7 on the left side. The picture on the right side gives the distribution of the significant wave height after two days simulation time.

After cloning the WAM repository to a local computer the full history with all previous versions and descriptions of it is available and can be used to work on. Detailed descriptions of all possibilities of the Git system are included in the book 'Pro Git' (Chacon, 2009) which is available online in the net under : http://git-scm.com/book .

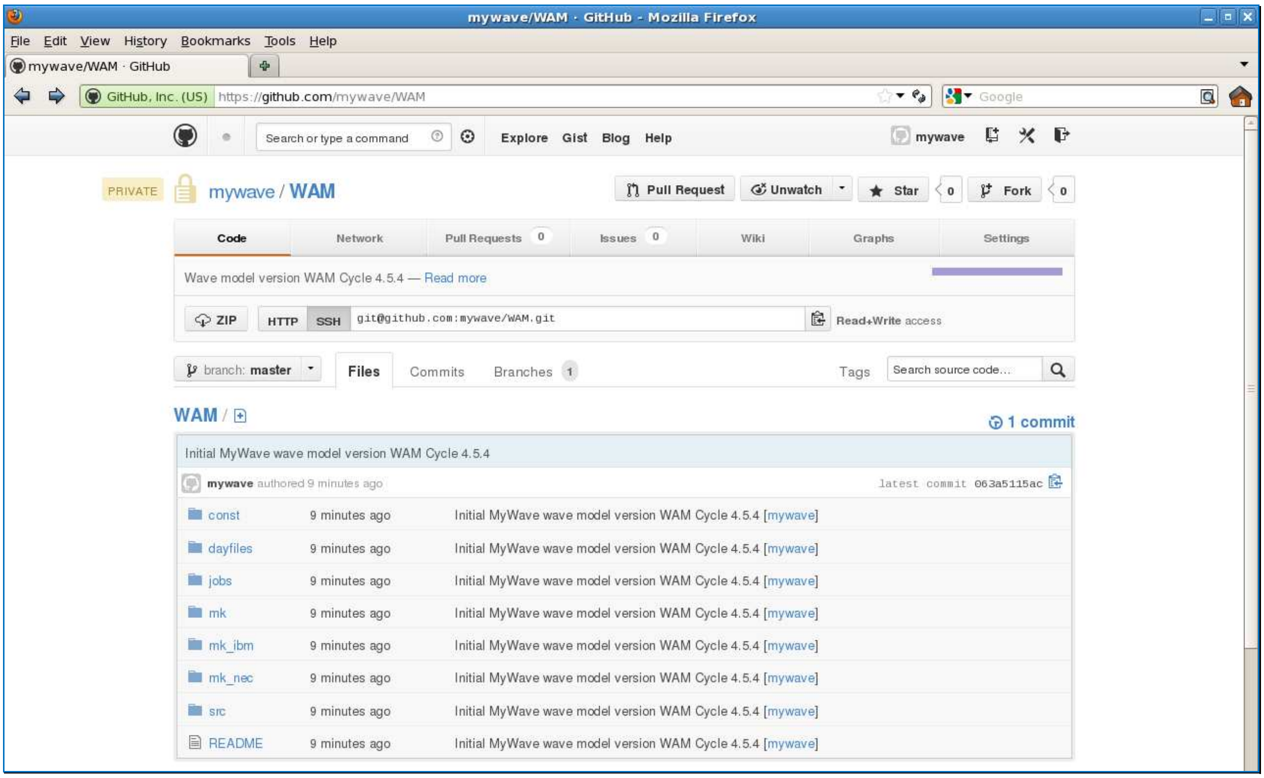

Figure 6: Contents of the WAM repository on the GitHub server

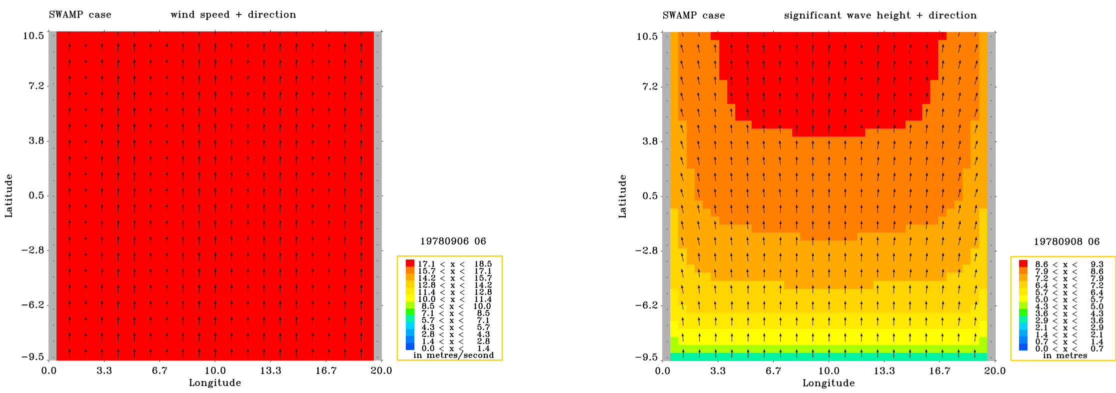

Figure 7: SWAMP case - distribution of wind (left) and significant wave height after two days (right)

To work with Git on a local computer is easy, usually two solutions are offered by the system. The first possibility is to manage the corresponding Git repository in the line command mode as given in an example in figure 8 that includes a Git command to generate a list of the source code modules combined with some information about the size of the individual modules. That list will be shown together with some general information of the WAM Git repository. Furthermore there is usually a graphical tool available called ‘Gitk’ which offers a very convenient method to manage the corresponding Git repository. Figure 9 includes an example of Gitk – in this case the last changes made in the WAM source code of the current master branch.

```
behrens@hpcsun-master:/data/behrens/mywave> git log --stat src/mod*

commit d034fd57697bd998427ea6acff1213d10bab04f9
Author: Arno Behrens <arno.behrens@hzg.de>
Date:   Fri Sep 21 14:29:53 2012 +0200

    initial MYWAVE version WAM Cycle 4.5.4

 src/mod/preproc_module.f90          | 1592 ++++++++++++++++++++++++
 src/mod/preproc_user_module.f90     |  353 ++++++
 src/mod/wam_assi_module.f90         | 2256 ++++++++++++++++++++++++++++++++++
 src/mod/wam_assi_set_up_module.f90  |  436 +++++++
 src/mod/wam_boundary_module.f90     |  755 ++++++++++++
 src/mod/wam_coldstart_module.f90    |  515 ++++++++
 src/mod/wam_coordinate_module.f90   |  943 ++++++++++++++
 src/mod/wam_current_module.f90      | 1281 +++++++++++++++++++
 src/mod/wam_file_module.f90         |  658 ++++++++++
 src/mod/wam_fre_dir_module.f90      |  572 +++++++++
 src/mod/wam_general_module.f90      | 1841 +++++++++++++++++++++++++++
 ...
```

Figure 8: Git management in line command mode

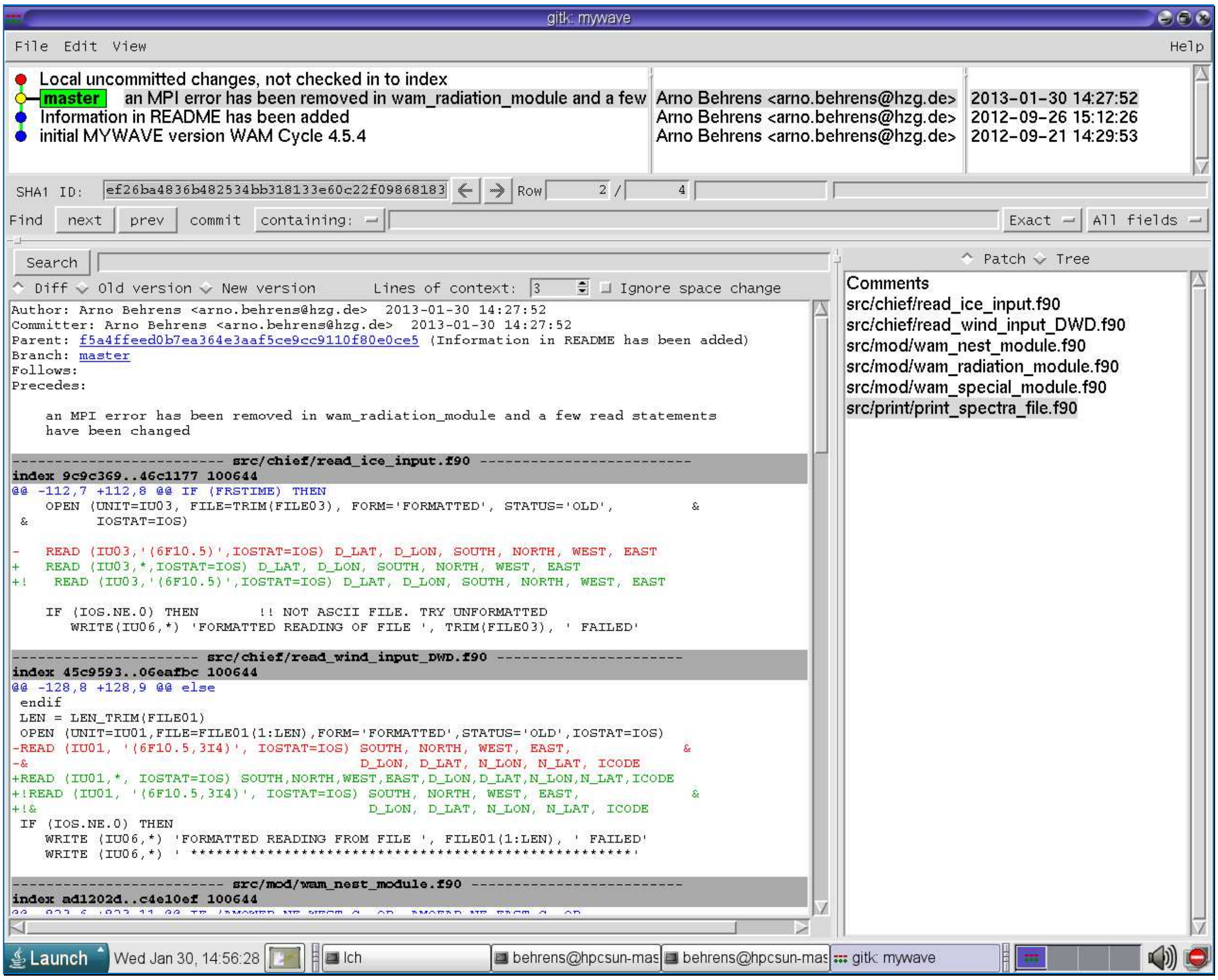

Figure 9: Example for working with the graphical tool Gitk

III WAM MANUAL

Within the last twenty years the WAM Cycle 4 wave prediction model has become a standard tool for operational wave predictions as well as for research and engineering applications. The model is widely distributed and used by more than 150 institutions. The availability of high-speed computers and the increasing demands for wave prediction products have led to this large user community of the model code. The quality of the wave analysis and forecasts are continuously improving, mainly due to a much better quality of the forcing wind fields. Only minor changes have been introduced into the model itself. This is a clear indication, that the approach taken twenty years ago by the WAM group has been very successful.

The model code developed in 1991 does not include any progress made in physics, numerics, and computer technology. On the other hand the code distribution has created a large user community with a wide range of applications for the model. Therefore it is an on-going task to take into account the progress made as well as the special demands of the wide user community of the model.

The new designed WAM Cycle 4.5.4 used in the MyWave project is an update of the WAM Cycle 4 wave model, which is described in Komen et al. 1994 and Guenther et al. 1992. Since the following chapters include descriptions of all the updates and extensions that have been made to improve the old WAM Cycle 4, the former manual (Günther et al., 1992) has been added to the WAM repository on the Github server. It explains the basic theory and the underlying equations.The basic physics and numerics are kept in the new release. The source function integration scheme made by Hersbach and Janssen, 1999, and the model up-dates (Bidlot, et al., 2005) are incorporated. The other main improvements introduced in WAM Cycle 4.5.4 are technical improvements, which take into account the new possibilities of Fortran 95. On request from the user community a number of additional options are added in the model. These changes are documented in Chapter III.1.

III.1 WAM Cycle 4.5.4 Updates and Extensions

III.1.1 Source Function Integration

The new method is semi implicit and based on the developments at ECMWF (Janssen, personal communication). It is

$$F_{n+1} = F_n + \frac{\Delta t S}{1 - \Delta t G}$$

Where $S = S(u_{n+1}, F_n)$ is the source function computed with the spectrum at time n and the wind speed at n+1. $G = G(u_{n+1}, F_n)$

The wave model dissipation source function has been reformulated in terms of a mean steepness parameter and a mean frequency that gives more emphasis on the high-frequency part of the spectrum and results in a more realistic interaction between wind sea and swell. This has allowed the relaxation of the prognostic frequency range over which the model equations are integrated. A few other small adjustments were also necessary to take advantage of the increased dynamic range of the model (Bidlot, et al., 2005).

III.1.2 Time Stepping

The main restriction for high-resolution applications of WAM Cycle 4 was, that time steps have to be a multiple of one minute. Therefore the time variables are extended by seconds. In addition the century is included. The new format is:

CHARACTER (LEN=14) 'YYYYMMDDHHMMSS'

The model now allows that the propagation time step is longer than the source function time step, which may speed up the computations for very high spatial resolution. The following restrictions for time steps must still be fulfilled:

All time steps must have integer ratios with the restart time step,
Source and propagation time step must have integer ratios with the wind input time step,
Output times must be integer multiples of the propagation time step.

See Annex A and D for details.

III.1.3 Sea Ice

If a file with a sea ice map is provided to the model, the wave spectra at all grid points marked as ice will be set to zero after a propagation has been done. Ice field can be changed during a model run.

See Annex A and B for details.

III.1.4 Output of Integrated Parameters

The user can select the following integrated parameters to be processed and stored as gridded fields in the output file:

Wind speed U_{10}, wind direction, friction velocity, drag coefficient, normalised wave stress, significant wave height, peak period, mean period, Tm1 period, Tm2 period, mean direction and mean spread. The wave parameters can be requested for the full wave spectrum, the wind sea spectrum and/or the swell spectrum.

See Annex A and D for details.

III.1.5 Output of Spectra

Output of spectra is possible at specified output sites. These sites are now defined in the *WAM_User* file of CHIEF instead of in the *Preproc_User* file. The full spectrum, the wind sea and/or the swell spectrum can be written to one output file and/or printed in the *WAM_Prot* file. The header of each spectrum contains all integrated parameters computed from the spectrum.

See Annex A and D for details.

III.1.6 Multiple Nests in Coarse Grid

A coarse grid run can generate output files for more than one nest.

See Annex A and C for details.

III.1.7 Input of Boundary Spectra in a Fine Grid Model Run

Time interpolation of boundary input spectra into a fine grid run is included in the SUBROUTINE BOUNDARY_INPUT (in WAM_BOUNDARY_MODULE) of the main program CHIEF. Therefore the old main program BOUINT is removed from the WAM system. (Carretero, personal communication). The user can control the output time step of boundary values.

See Annex A and C for details.

III.1.8 Angular Directions

The spectral directions are turned by half of a direction increment to avoid directions parallel to the grid axis. This results in a better propagation performance.

III.1.9 Blocking

The option for a blocked grid computation is removed.

III.1.10 PRESET Program

The PRESET program has been removed from the WAM software. The coldstart options are moved into the main Program CHIEF.

III.1.11 Depth Induced Wave Breaking

Optional depth induced wave breaking (Battjes & Janssen, 1978) has been included as an additional source function. The code has been taken from the Promise version of WAM.

III.1.12 In-stationary Current and Water Depth

Optionally the currents and /or the water depth can be changed during a model run. This includes that sea points can become dry.

See Annex A and B for details.

III.1.13 Output of Radiation Stress, wave force and Stokes Drift

Optional radiation stresses are computed during a model run and saved into a separate file or printed into the *WAM_Prot* file.

See Annex A and F for details.

III.1.14 Namelist Formatted Control Parameters

Namelists can be used for all main programs instead of the formatted user files to define the control parameters.

See Annex A for details.

III.1.15 Input of coordinates, grid increments and internal representation.

The use of high spatial resolution grids was limited in the old WAM versions, because all coordinates where defined as real values in degrees. Therefore the internal representation of coordinates has been change to integer values in 100*seconds. This enables the program to handle grids with a minimum resolution of 0.02 seconds, which corresponds to about 1m.

III.1.16 Reduced Gaussian Grid

An option to generate a reduced Gaussian grid was introduced.

See Annex G for details.

III.2 WAM Cycle 4.5.4 Source Code

The whole program system is coded in standard FORTRAN 90.

The main features used are:

- ‘Free format’ code,
- Modules instead of common blocks,
- Dynamical allocation of arrays,
- Application of new intrinsic functions,
- IMPLICIT none,
- INTERFACE blocks,
- USE module, ONLY.

This implies, that one executable can be applied for all applications (parameter statements for array dimensions have not to be changed anymore). The use of ‘IMPLICIT NONE’, ‘INTERFACE’ and ‘USE module, ONLY’ was introduced to make the code more robust and to prevent coding error.

Table 1: WAM source code modules

Module	used by	Purpose
preproc_module	PREPROC	Stores variables and routines special to PREPROC
preproc_user_module	PREPROC	Default setting and user input of PREPROC control parameters
wam_boundary_module	CHIEF	Input and output of nest boundary values
wam_coldstart_module	CHIEF	Generates cold start
wam_coordinate_module	all programs	Processes model coordinates
wam_current_module	CHIEF	Stores and processes currents
wam_file_module	all programs	Stores definition of input, output, and file names and units
wam_fre_dir_module	PREPROC, CHIEF	Stores frequency-direction data and routines for processing
wam_general_module	all programs	Basic subroutines and functions
wam_grid_module	PREPROC, CHIEF	Stores model grid
wam_ice_module	CHIEF	Stores and processes ice fields
wam_initial_module	CHIEF	Routines and variables to initialise the model
wam_interface_module	CHIEF	Routines to process spectra
wam_model_module	CHIEF	Run-time model data (spectra, wind, currents, depth)
wam_nest_module	PREPROC, CHIEF	Stores nest information and routines to generate them
wam_output_module	CHIEF	Performs output of integrated parameters and spectra
wam_output_set_up_module	CHIEF	Stores scaling information, parameter names, output options, and output sites
wam_print_module	post processing	Stores variables and routines used by the post-processing programs
wam_print_user_module	post processing	Default setting and user input of post-processing control parameters
wam_propagation_module	CHIEF	Propagation computation
wam_radiation_module	CHIEF	Stores routines to compute and to write and radiation stresses
wam_restart_module	CHIEF	Generates hot start and saves restart files
wam_source_module	CHIEF	Source function computation
wam_timopt_module	all programs	Store model times and options
wam_topo_module	CHIEF	Stores and processes water depth fields
wam_user_module	CHIEF	Default setting and user input of CHIEF control parameters
wam_wind_module	CHIEF	Stores and processes winds

Remark:

Some modules are interacting. Therefore some compilers require a specific order for the compilation (see Chapter III.5). All modules must anyhow be compiled before the other program units.

III.3 The Model System

The model system consists of three major program parts:

1. Pre-processing program
2. Processing program
3. Post-processing programs

The WAM model is designed to run as a module of a more general system (atmosphere/waves or currents/waves) or as a stand-alone program.

See Annex A to F for details of user control parameters and user input files.

III.3.1 Pre-processing Program

PREPROC generates time independent information for the wave model. Starting from a regional or global topographic data set, the model grid is created in the form required for the model. The frequency and angular arrays are generated.

A number of model constants are pre-computed and stored together with the model grid, frequency, and angular information in the output file.

If nested grids are generated, the information for the output, input and interpolation of boundary spectra are pre-computed.

A topographic data file has to be provided by the user. If a fine grid run is requested, the PREPROG output file from the coarse grid is necessary, too.

III.3.2 Processing Program

CHIEF is the shell program of the stand-alone version of the wave model calling the subroutine version of the wave model. All time dependent variables and user-defined parameters are fixed, the wind fields are transformed into the model formats, and the transport equation is integrated over a chosen period. The initial spectra are generated in case of a cold start.

The program uses the output file of PREPROC as set-up file. A wind input file and optional ice file and/or current file and/or water depth file and/or boundary value files have to be provided by the user.

The user can select a number of model options and parameters. The following model options are implemented:

- Cartesian or spherical propagation,
- Deep or shallow water,
- Without or with depth or with depth and current refraction,
- Depth induced breaking,
- Nested grids,
- Time interpolation of winds, currents, water depth, and ice fields or no time interpolation,
- Model output at regular intervals or by list,
- Printer and/or file output of individually selected parameters,
- Output variables,
- Output sites for spectra,
- Cold or hot start.

The model results (if selected) are saved in three files, one for integrated parameters (MAP…), one for spectra (OUT…) at specified sites and one for radiation stresses (RAD…).

III.3.3 Post-processing Programs

Four post-processing programs are provided:

1. PRINT_GRID_FILE: Prints the maps of integrated parameters,
2. PRINT_TIME: Prints time series of integrated parameters at selected sites,
3. PRINT_SPECTRA_FILE: Prints time series of spectra at selected output sites,
4. PRINT_RADIATION_FILE: Prints the maps of radiation stress parameters.

The programs are set up for the model result files. Controlled by the user input the results are printed. Plot software is not included in the standard set of programs.

III.4 Communication between the Sub Systems

The program system uses 6 different types of files:

- User input files, which are needed by each program to control the execution.
- Protocol output files, which are generated by each program.
- Input data files, which have to be provided by the user.
- A Set-up file, which is generated by PREPROC and used by CHIEF.
- Result files, which are generated by CHIEF and used by the post-processing programs.
- Restart files, which are generated and used by CHIEF.

Figures 10, 11 and 12 show an overview about the input and output files used by the different main programs.

The file names for the **user input and protocol output files** are defined in the modules and cannot be changed from outside the program. The files have to be in the directory where the program is executed. The user input files have a fixed format or are namelists. Examples are provided with the code. See Annex A for details.

Input data files are:

- Topographic data for PREPROC,
- Wind data for CHIEF,
- Current data for CHIEF (optional),
- Depth data for CHIEF (optional),
- Ice data for CHIEF (optional).

These files are dynamically assigned by OPEN. The file names must be defined in the user input files. The full path names have to be provided if the data are not in the directory where the program is executed. See Annex B for details.

Set-up file is generated by PREPROC. It contains the model constants and the general grid information. This file is dynamically assigned by OPEN. The file names are pre-defined in the user modules, but can be changed in the user input files. The full path names have to be provided if the data are not in the directory where the program is executed. The set-up file is unformatted.

Result files are the model output files generated by CHIEF. Different files store the integrated data, the spectra and the radiation stress output. If the nesting option is on the model generates boundary value files for a follow-up fine grid or reads in boundary spectra from existing files. All these files are dynamically assigned by OPEN. The file names are built from in the user modules pre-defined file identifier, which can be changed in the user input files, extended by the date of the last output stored in the file. The full path names have to be provided if the data are not stored in the directory where the program is executed. Details of the file name convention are given in Subroutine OPEN_FILE, which is located in the WAM_GENERAL_MODULE. All result files are unformatted. **Restart files** follow the same rules as result files.

Fortran read and write units inside the programs are integer variables following the convention IUxx, where xx is the unit number, e.g. xx = 01, xx = 11. The default units and standard filenames are defined in the user modules and can be changed in the user input files.

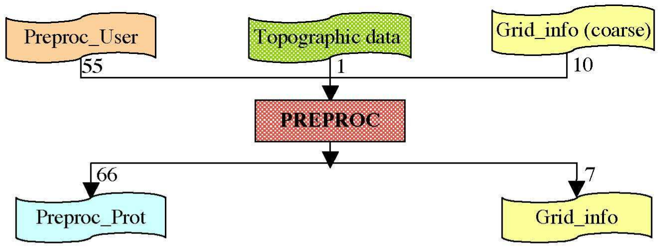

Figure 10: Input and output files for PREPROC

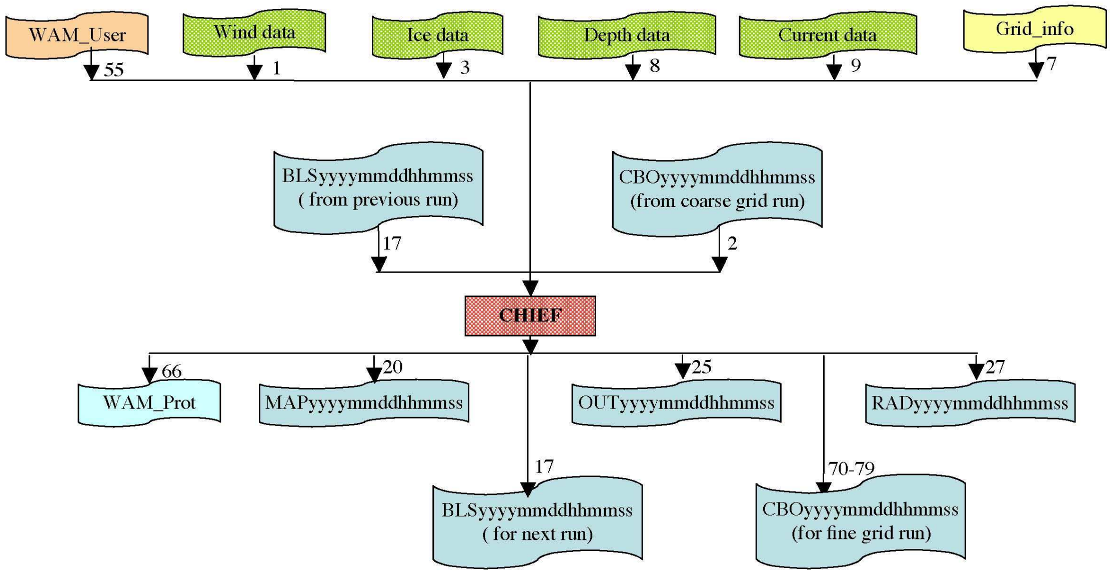

Figure 11: Input and output files for CHIEF

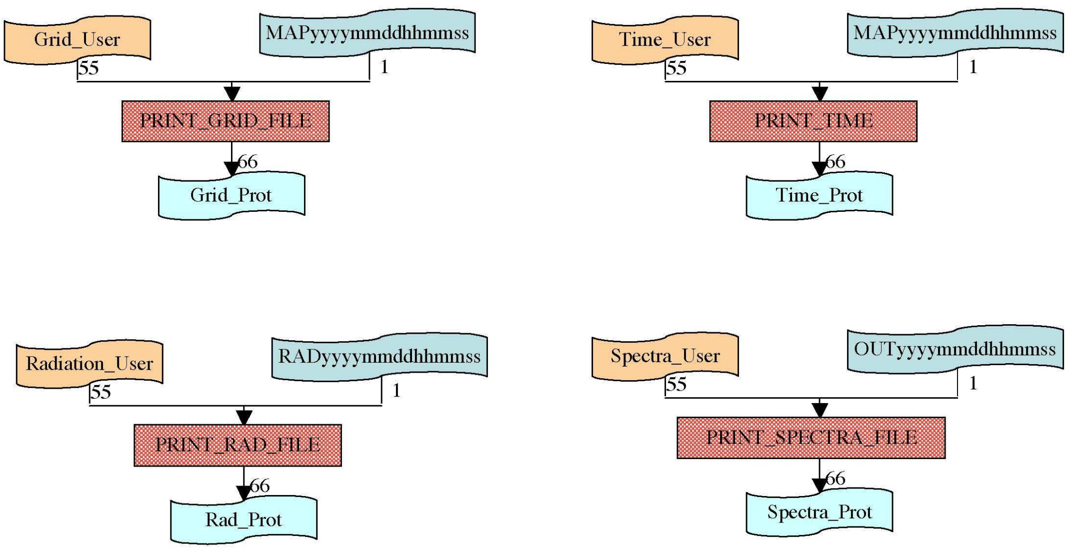

Figure 12: Input and output files for the post-processing programs

III.5 Compile Order for Modules

The modules provided with the model code are interdependent. Therefore the modules and programs have to be compiled in the following order:

III.5.1 Pre-processing program PREPROC

WAM_SOURCE/Module/WAM_FILE_MODULE.f90
WAM_SOURCE/Module/WAM_COORDINATE_MODULE.f90
WAM_SOURCE/Module/WAM_GENERAL_MODULE.f90
WAM_SOURCE/Module/WAM_FRE_DIR_MODULE.f90
WAM_SOURCE/Module/WAM_GRID_MODULE.f90
WAM_SOURCE/Module/WAM_NEST_MODULE.f90
WAM_SOURCE/Module/PREPROC_MODULE.f90
WAM_SOURCE/Module/PREPROC_USER_MODULE.f90
WAM_SOURCE/Preproc/PREPROC.f90
WAM_SOURCE/Preproc/READ_TOPOGRAPHY.f90
WAM_SOURCE/Preproc/READ_PREPROC_USER.f90

III.5.2 Processing program CHIEF

WAM_SOURCE/Module/WAM_FILE_MODULE.f90
WAM_SOURCE/Module/WAM_COORDINATE_MODULE.f90
WAM_SOURCE/Module/WAM_GENERAL_MODULE.f90
WAM_SOURCE/Module/WAM_TIMOPT_MODULE.f90
WAM_SOURCE/Module/WAM_FRE_DIR_MODULE.f90
WAM_SOURCE/Module/WAM_MODEL_MODULE.f90
WAM_SOURCE/Module/WAM_INTERFACE_MODULE.f90
WAM_SOURCE/Module/WAM_GRID_MODULE.f90
WAM_SOURCE/Module/WAM_TOPO_MODULE.f90
WAM_SOURCE/Module/WAM_CURRENT_MODULE.f90
WAM_SOURCE/Module/WAM_ICE_MODULE.f90
WAM_SOURCE/Module/WAM_OUTPUT_SET_UP_MODULE.f90
WAM_SOURCE/Module/WAM_WIND_MODULE.f90
WAM_SOURCE/Module/WAM_NEST_MODULE.f90
WAM_SOURCE/Module/WAM_BOUNDARY_MODULE.f90
WAM_SOURCE/Module/WAM_SOURCE_MODULE.f90
WAM_SOURCE/Module/WAM_OUTPUT_MODULE.f90
WAM_SOURCE/Module/WAM_PROPAGATION_MODULE.f90
WAM_SOURCE/Module/WAM_RADIATION_MODULE.f90
WAM_SOURCE/Module/WAM_COLDSTART_MODULE.f90
WAM_SOURCE/Module/WAM_RESTART_MODULE.f90
WAM_SOURCE/Module/WAM_INITIAL_MODULE.f90
WAM_SOURCE/Module/WAM_USER_MODULE.f90
WAM_SOURCE/Chief/CHIEF.f90
WAM_SOURCE/Chief/WAVEMDL.f90
WAM_SOURCE/Chief/INITMDL.f90
WAM_SOURCE/Chief/WAMODEL.f90
WAM_SOURCE/Chief/PRINT_WAM_STATUS.f90
WAM_SOURCE/Chief/READ_WAM_USER.f90
WAM_SOURCE/Chief/READ_WIND_INPUT.f90
WAM_SOURCE/Chief/READ_TOPO_INPUT.f90
WAM_SOURCE/Chief/READ_CURRENT_INPUT.f90
WAM_SOURCE/Chief/READ_BOUNDARY_INPUT.f90
WAM_SOURCE/Chief/READ_ICE_INPUT.f90

III.5.3 Post-Processing program PRINT_GRID_FILE

WAM_SOURCE/Module/WAM_FILE_MODULE.f90
WAM_SOURCE/Module/WAM_COORDINATE_MODULE.f90
WAM_SOURCE/Module/WAM_GENERAL_MODULE.f90
WAM_SOURCE/Module/WAM_PRINT_MODULE.f90
WAM_SOURCE/Module/WAM_PRINT_USER_MODULE.f90
WAM_SOURCE/Module/WAM_OUTPUT_SET_UP_MODULE.f90
WAM_SOURCE/Module/WAM_PRINT_MODULE.f90
WAM_SOURCE/Print/PRINT_GRID_FILE.f90
WAM_SOURCE/Print/READ_GRID_FILE.f90
WAM_SOURCE/Print/READ_GRID_USER.f90

III.5.4 Post-Processing program PRINT_TIME

WAM_SOURCE/Module/WAM_FILE_MODULE.f90
WAM_SOURCE/Module/WAM_COORDINATE_MODULE.f90
WAM_SOURCE/Module/WAM_GENERAL_MODULE.f90
WAM_SOURCE/Module/WAM_TIMOPT_MODULE.f90
WAM_SOURCE/Module/WAM_PRINT_MODULE.f90
WAM_SOURCE/Module/WAM_PRINT_USER_MODULE.f90
WAM_SOURCE/Print/PRINT_TIME.f90
WAM_SOURCE/Print/READ_GRID_FILE.f90
WAM_SOURCE/Print/READ_TIME_USER.f90

III.5.5 Post-Processing program PRINT_SPECTRA_FILE

WAM_SOURCE/Module/WAM_FILE_MODULE.f90
WAM_SOURCE/Module/WAM_COORDINATE_MODULE.f90
WAM_SOURCE/Module/WAM_GENERAL_MODULE.f90
WAM_SOURCE/Module/WAM_PRINT_MODULE.f90
WAM_SOURCE/Module/WAM_PRINT_USER_MODULE.f90
WAM_SOURCE/Print/PRINT_SPECTRA_FILE.f90
WAM_SOURCE/Print/READ_SPECTRA_FILE.f90
WAM_SOURCE/Print/READ_SPECTRA_USER.f90

III.5.6 Post-Processing program PRINT_RADIATION_FILE

WAM_SOURCE/Module/WAM_FILE_MODULE.f90
WAM_SOURCE/Module/WAM_COORDINATE_MODULE.f90
WAM_SOURCE/Module/WAM_GENERAL_MODULE.f90
WAM_SOURCE/Module/WAM_PRINT_MODULE.f90
WAM_SOURCE/Module/WAM_PRINT_USER_MODULE.f90
WAM_SOURCE/Module/WAM_OUTPUT_SET_UP_MODULE.f90
WAM_SOURCE/Print/PRINT_RADIATION_FILE.f90
WAM_SOURCE/Print/READ_RADIATION_FILE.f90
WAM_SOURCE/Print/READ_RADIATION_USER.f90

III.6 Model Flow Diagrams

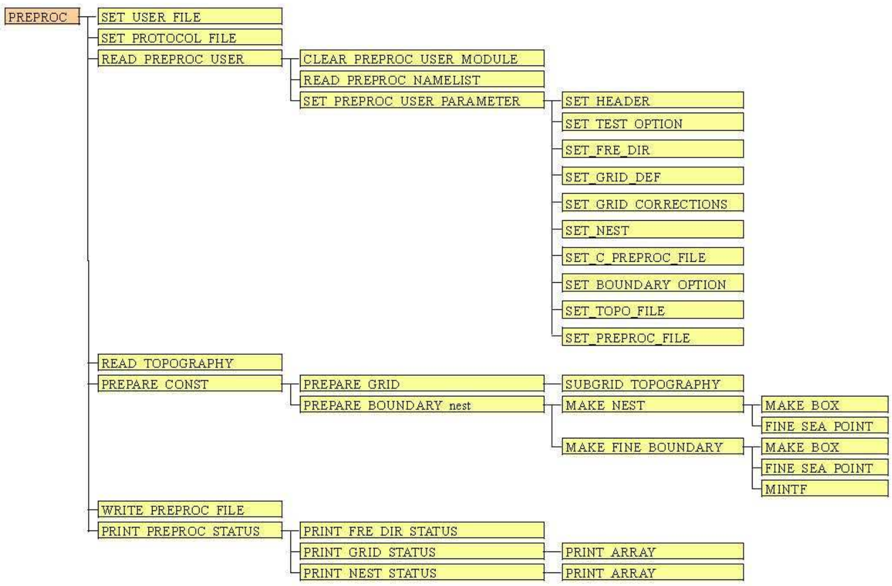

Figure 13: Flow diagram of main program PREPROC

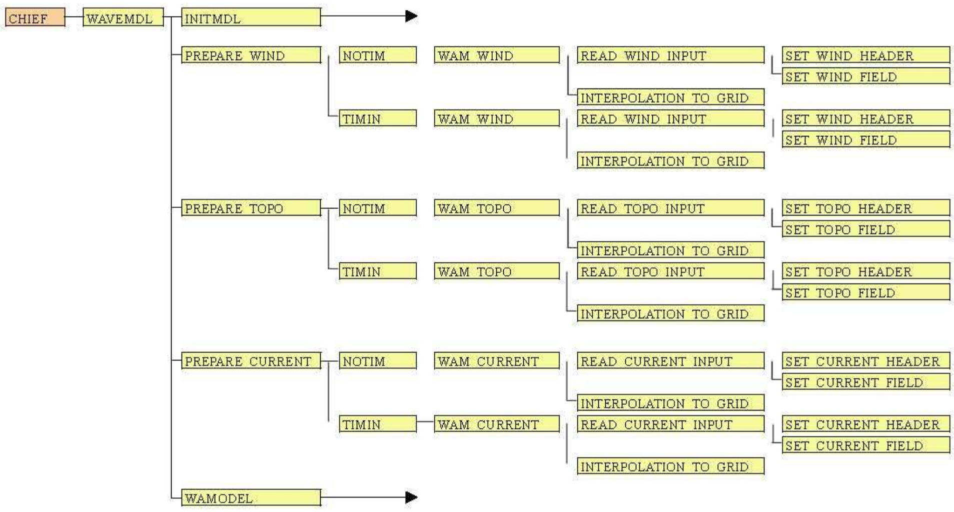

Figure 14: Flow diagram of main program CHIEF

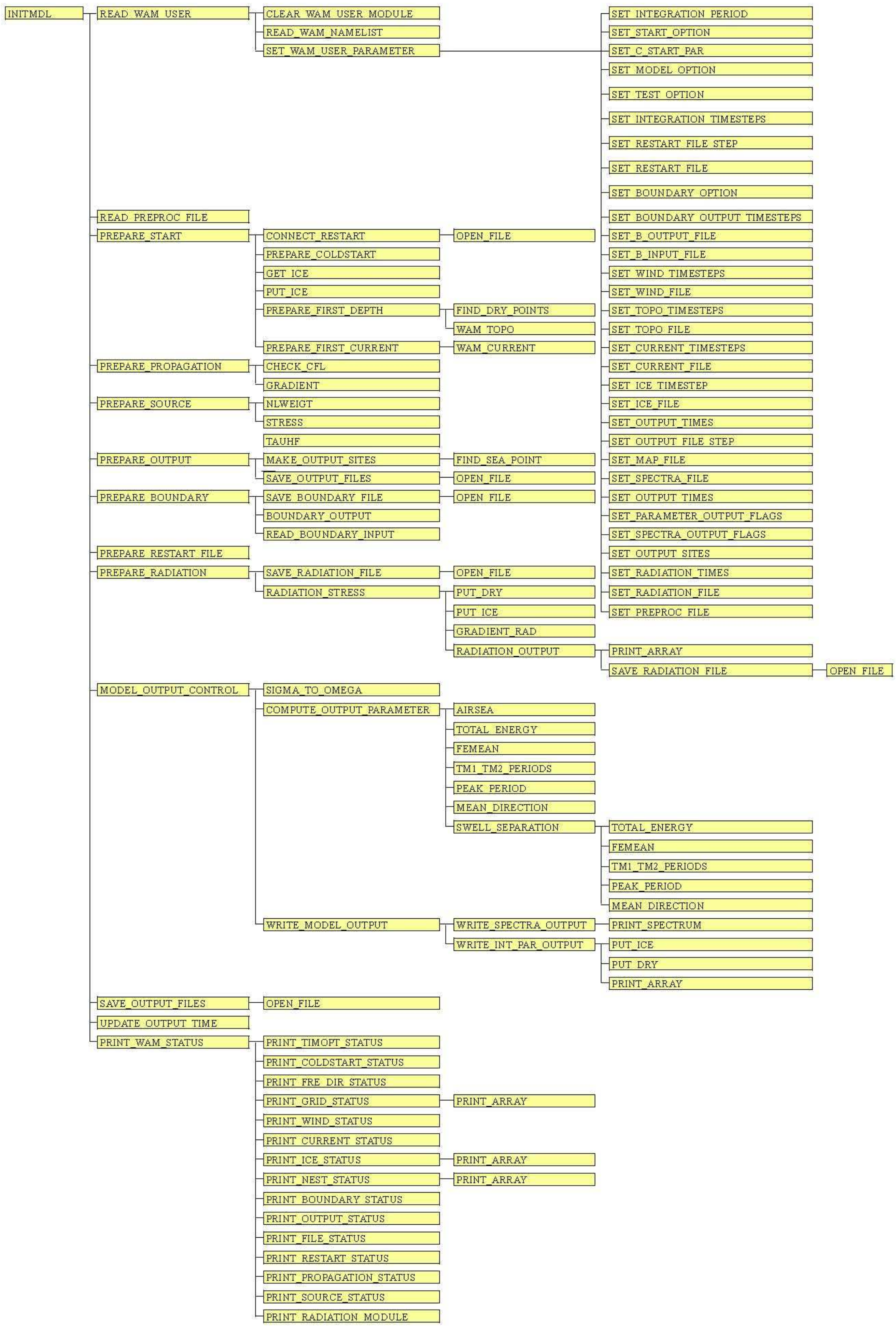

Figure 15: Flow diagram of subroutine INITMDL of main program CHIEF

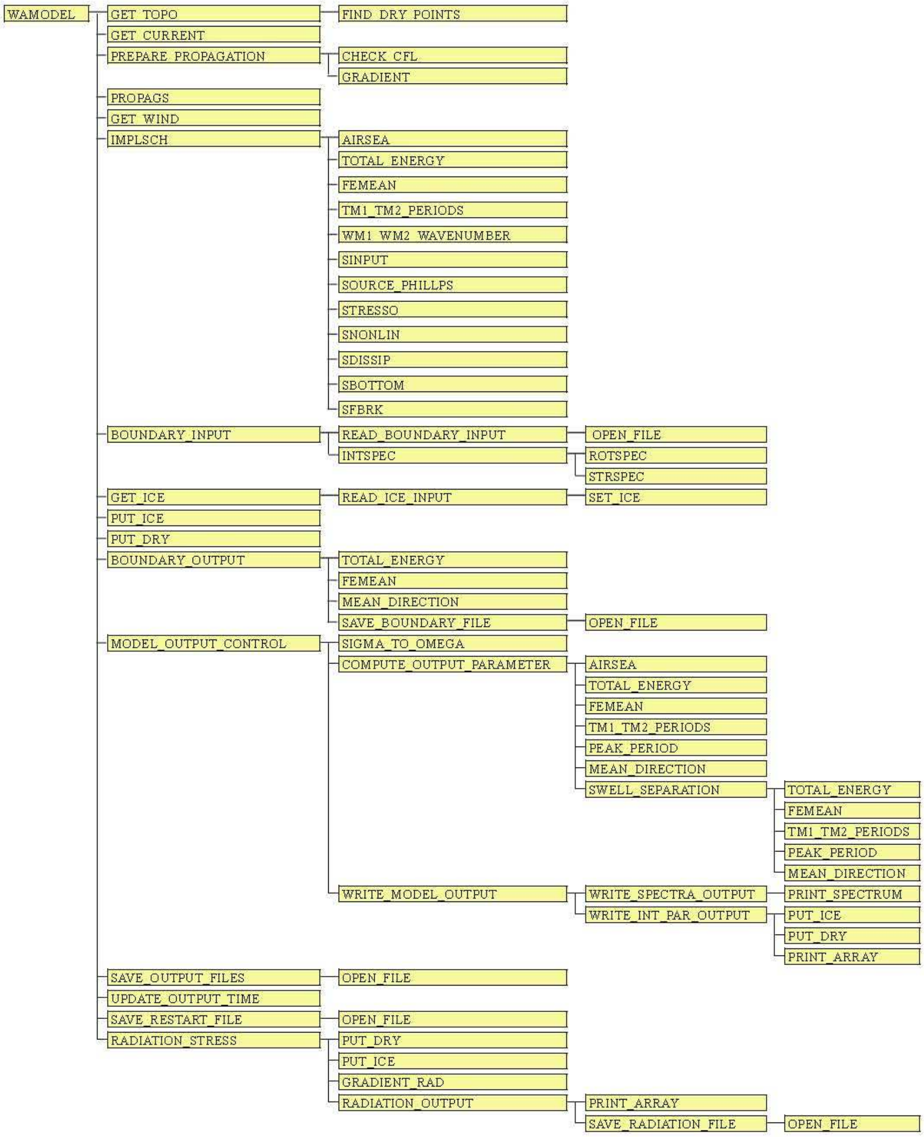

Figure 16: Flow diagram of subroutine WAMODEL of main program CHIEF

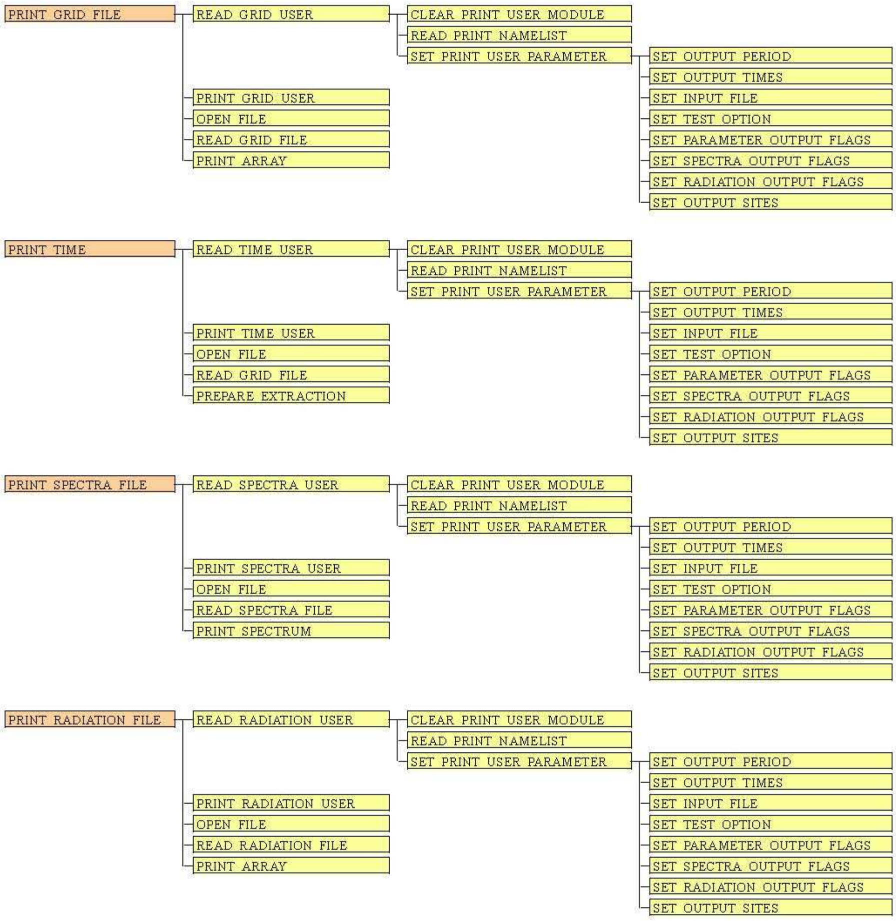

Figure 17: Flow diagram of the main post-processing program

IV SUMMARY AND OUTLOOK

According to the specification agreed upon in the work package description of the MyWave project, all the work for task 1.4 of WP1 has been done successfully. The web-based source code library for the MyWave version WAM Cycle 4.5.4 is available in the web in due time on the Github server (http://mywave.github.io/WAM/) and can be accessed by all registered MyWave contributors. The WAM repository includes a complete set-up for one of the SWAMP cases and can be downloaded to an arbitrary local computer to work with it. New developments achieved during MyWave can be individually inserted into a local WAM repository, but the master repository on the Github server should be changed by the account owner only to avoid confusion and to make sure that only fully tested and checked branches will be included in the master WAM repository. During the lifespan of MyWave the WAM repository will remain private, but afterwards the new developed MyWave version will be accessible for the general user in a public repository including all the new developments achieved during the MyWave project.

V REFERENCES

Battjes, J.A., Janssen, J.P.F.M., 1978. Energy loss and set-up due to breaking of random waves. In: Proceedings of the 16th Conference on Coastal Engineering, ASCE, Hamburg, Germany, vol. 1, pp. 569–587

Chacon, S., 2009: Pro Git, Apress, ISBN-13 **:** 978-1430218333

Günther, H., S. Hasselmann, P.A.E.M. Janssen, 1992: The WAM Model Cycle 4.0. User Manual. Technical Report No. 4, Deutsches Klimarechenzentrum, Hamburg, Germany. 102 pages.

Hersbach, H. and P.A.E.M. Janssen, 1999: Improvements of the short fetch behavior in the WAM model. J. Atmos. Oceanic Techn., 16, 884-892.

Komen, G.J., L.Cavaleri, M. Donelan, K. Hasselmann, S. Hasselmann and P.A.E.M. Janssen, 1994: Dynamics and modelling of ocean waves. Cambridge University Press, Cambridge, UK, 560 pages.

Bidlot, J., P. Janssen and S. Abdalla, 2005: A revised formulation for ocean wave dissipation in CY29R1. MEMORANDUM RESEARCH DEPARTMENT of ECMWF, April 7, 2005 File: R60.9/JB/0516

SWAMP group, 1985: Ocean wave modeling, Plenum Press, New York and London

VI ANNEX A – USER INPUT :

VI.1 Introduction

This Annex describes the input parameters, which have to be defined to control the different options and settings in the WAM-Model programs.

VI.2 Concept of user input

All control parameters, which can or must be defined by the user, are combined in NAME-LISTs. NAMELISTs are defined for PREPROC, the main WAM model CHIEF, and the post-processing programs. The list is read either from a file or from standard input. Alternatively formatted text files can be used as input for the most important control parameters.

The programs will look for a file. If the file exists, it will first try to read the NAMELIST, if this fails a formatted reading is used for the formatted text file. If a file does not exist, the NAMELIST must be in the standard input.

Examples of the NAMELISTs and formatted text files are provided with the code.

VI.3 PREPROC Control Parameters

The code for the PREPROC program is in PREPROC_USER_MODULE and in READ_PREPROC_USER.

The program tries to open a file with the name

'Preproc_User'.

If the file exists it will try to read the PREPROC_NAMELIST. If the reading was not successful, the program will read the same file in a formatted style as required by the program READ_PREPROC_USER. If the file does not exist the PREPROC_NAMELIST is read from standard input.

Default values for the control parameter are defined in subroutine CLEAR_PREPROC_USER_MODULE in the PREPROC_USER_MODULE.

The NAMELIST variables are listed in Tab. 2. The user has to define the variables marked by "-user-" to assure a successful PREPROC run for a grid set-up without nesting and depth corrections.

The model grid axes are defined by the parameters marked by "-user-[3] ". If none of the parameters is given the axis definitions will be taken from the header included in the topography input file. Only 3 out of 4 parameters have to be defined. E.g. for the longitudes possible combinations are: (`NX, XDELLO, AMOWEP`) or (`NX, AMOWEP, AMOEAP`) or (`XDELLO, AMOWEP, AMOEAP`).

The water depth can be changed in up to 80 areas. The boundaries of areas must be defined by the parameter arrays `XOUTS, XOUTN, XOUTW, XOUTE` and the new depth in `NOUTD`. If the water depth in more than 80 areas has to be changed, the parameter `NOUT` in the PREPROC_USER_MODULE must be updated.

For a coarse grid set-up the up to 20 nest areas can be defined. The first elements of the arrays `AMOSOC, AMONOC, AMOWEC` and `AMOEAC` must be filled with the nest boundary coordinates. In addition a name can be defined for each nest area. If more than 20 nest areas have to be defined, the parameter `N_NEST` in the PREPROC_USER_MODULE must be increased.

All coordinates and grid increments are CHARACTER (LEN=13) variables. The input can be either in REAL degrees, formatted as F13.8, or formatted as 'sDDD:MM:SS.SS',

where

s is the sign of the coordinate (+ for East and North or – for West and South),
DDD are the degrees,
MM are the minutes and
SS.SS are the seconds of the coordinate (smallest value allowed is 00.02).

For a fine grid set-up the parameter `PREPROC_C_INPUT_FILE_NAME` must be defined with the coarse grid PREPROC output file name. If a file name is not given or the file does not exist a fine grid is not set-up.

Table 2: PREPROC_NAMELIST

[1] C(l) = CHARACTER (LEN=l); R = REAL; I = INTEGER; L = LOGICAL, R(n) and I(n) arrays of dimension n, C(l, n) are CHARACTER arrays of (LEN=l) and dimension n
[2] –user- = parameter must be defined by user; -999 = undefined.
[3] see text for explanation.

Variable name	Type[1]	Default[2]	Description
HEADER	C(80)	'blank'	Name of the processed grid
ITEST	I	0	Test output level
Frequency direction grid			
KL	I	24	No. of wave directions
ML	I	25	No. of wave frequencies
FR1	R	0.04177248	First frequency
Basic model grid			
REDUCED_GRID	L	.FALSE.	If .TRUE. a reduced Gaussian grid is set-up; otherwise a regular spherical grid is generated
NX	I	-user-[3]	No. of latitudes in grid
NX	I	-user-[3]	No. of latitudes in grid
NY	I	-user-[3]	No. of longitudes in grid
XDELLA	C(13)	-user-[3]	Latitude increment
XDELLO	C(13)	-user-[3]	Longitude increment
AMOSOP	C(13)	-user-[3]	Most southern latitude of grid
AMONOP	C(13)	-user-[3]	Most northern latitude of grid
AMOWEP	C(13)	-user-[3]	Most western longitude of grid
AMOEAP	C(13)	-user-[3]	Most eastern longitude of grid
LAND	R	0	Depth greater than LAND are sea points (maybe dry sea points)
Depth correction areas in basic model grid			
XOUTS	C(13,80)	'blank'	Most southern latitudes of grid correction areas
XOUTN	C(13,80)	'blank'	Most northern latitudes of grid correction areas
XOUTW	C(13,80)	'blank'	Most western longitudes of grid correction areas
XOUTE	C(13,80)	'blank'	Most eastern longitudes of grid correction areas
NOUTD	R(80)	-999.	Depth in grid correction areas
Nest areas in a coarse Grid			
AMOSOC	C(13,20)	'blank'.	Most southern latitudes of nest areas for a coarse grid set-up
AMONOC	C(13,20)	'blank'	Most northern latitudes of nest areas for a coarse grid set-up
AMOWEC	C(13,20)	'blank'	Most western longitudes of nest areas for a coarse grid set-up
AMOEAC	C(13,20)	'blank'	Most eastern longitudes of nest areas for a coarse grid set-up
NEST_NAME	C(20,20)	'blank'	Names given to the nests for a coarse grid set-up
nestcode	I(20)	0	If == 1 then boundary values are written in ASCII otherwise output is binary

Fine grid set-up information from coarse grid preproc			
PREPROC_C_INPUT_FILE_UNIT	I	10	Logical unit number to read the coarse grid preproc output file for a fine grid set-up
PREPROC_C_INPUT_FILE_NAME	C(80)	'blank'	Name of the coarse grid preproc output file for a fine grid set-up
Depth data file for basic model grid			
TOPO_INPUT_FILE_UNIT	I	1	Logical unit number to read the depth data file.
TOPO_INPUT_FILE_NAME	C(80)	-user-	Name of the depth data file.
Preproc output file			
PREPROC_OUTPUT_FILE_UNIT	I	7	Logical unit number to write the preproc output file.
PREPROC_OUTPUT_FILE_NAME	C(80)	Grid_info	Name of the preproc output file.

VI.4 Main WAM model Control Parameters

The code for the main WAM-model program is in WAM_USER_MODULE and READ_WAM_USER.

The program tries to open a file with the name

'Chief_User'

If the file exists it will try to read the WAM_NAMELIST. If the reading was not successful, the program will read the same file in a formatted style as required by the program READ_WAM_USER. If the file does not exist the WAM_NAMELIST is read from standard input.

The NAMELIST variables are listed in Tab. 3-6. The user has to define the variables marked by "-user-" to assure a successful WAM run.

Default values for the control parameter are defined in subroutine CLEAR_WAM_USER_MODULE in the WAM_USER_MODULE.

If more than 100 fixed output dates or more than 20 output sites should be processed, the parameters `NOUTT` or `MOUTP` in the WAM_USER_MODULE must be increased.

All coordinates and grid increments are CHARACTER (LEN=13) variables. The input can be either in REAL degrees, formatted as F13.8, or formatted as 'sDDD:MM:SS.SS',

where

s is the sign of the coordinate (+ for East and North or – for West and South),
DDD are the degrees,
MM are the minutes and
SS.SS are the seconds of the coordinate (smallest value allowed is 00.02).

Table 3: WAM_NAMELIST (part1)

[1]C(l) = CHARACTER (LEN=l); R = REAL; I = INTEGER; L = LOGICAL, R(n) and I(n) arrays of dimension n, C(l, n) are CHARACTER arrays of (LEN=l) and dimension n [2] –user- = parameter must be defined by user; -999 = undefined.			
Variable name	Type[1]	Default[2]	Description
START_DATE	C(14)	-user-	Start date / time group of model run (YYYYMMDDhhmmss)
END_DATE	C(14)	-user-	End date / time group of model run (YYYYMMDDhhmmss)
Start option and cold start settings			
COLDSTART	L	.TRUE.	Model start option: True = cold-start; false = hot-start.
IOPTI	I	1	Cold-start option : 0: wind independent initial values. Jonswap spectrum with cosine square angular spreading from given parameters; 1: Jonswap spectrum with cosine square angular spreading from Konswap fetch laws; 2: Jonswap spectrum with cosine square angular

			spreading from Jonswap fetch laws. Energy from given parameters, if wind speed is zero
ALPHA	R	0.018	Phillips parameter of Jonwap spectra for cold-start
FM	R	0.200	Peak frequency of Jonwap spectra for cold-start
GAMMA	R	3.000	Over shoot factor of Jonwap spectra for cold-start
SIGMA_A	R	0.070	Left peak width of Jonwap spectra for cold-start
SIGMA_B	R	0.090	Right peak width of Jonwap spectra for cold-start
THETAQ	R	0.	Mean wave direction of spreading function for cold-start
FETCH	R	30000.	Fetch used for fetch laws for cold-start spectra [m]. If FETCH is not positive, then 0.5 of the latitude increment is used
		Model options	
SPHERICAL_RUN	L	.TRUE.	Propagation option: True: model runs on spherical grid; False: model runs on Cartesian grid
SHALLOW_RUN	L	.TRUE.	Shallow water option: True: model takes water depth into account; False: deep water run
REFRACTION_D_RUN	L	.FALSE.	Depth refraction option: True: depth refraction terms are used; False: depth refraction terms are ignored
REFRACTION_C_RUN	L	.FALSE.	Current refraction option: True: current refraction terms are used; False: current refraction terms are ignored
WAVE_BREAKING_RUN	L	.FALSE.	Depth induced wave breaking option: True: source term is active; False: source term is ignored
PHILLIPS_RUN	L	.FALSE.	Phillips linear growth option: True: source term is active; False: source term is ignored
ITEST	I	0	Test output level
		Integration time steps	
PROPAGATION_TIMESTEP	I	-user-	Propagation integration time step
PROPAGATION_TIMESTEP_UNIT	C(1)	'S'	Unit of propagation time step (M = minute, H = hour, S = second)
SOURCE_TIMESTEP	I	0	Source function integration time step <= 0: source function = propagation time step
SOURCE_TIMESTEP_UNIT	I	'S'	Unit of source function integration time step (M = minute, H = hour, S = second)
		Restart settings	
RESTART_SAVE_TIMESTEP	R	0	Time step to save restart >0: restart files are saved in regular time steps =0: restart file is saved at the end of the run <0: restart file is not saved
RESTART_SAVE_TIMESTEP_UNIT	C(1)	'S'	Unit of time step to save restart files (M = minute, H = hour, S = second)
RESTART_FILE_UNIT	I	17	Logical file unit number to read and write restart file
RESTART_FILE_NAME	C(80)	'BLS'	File identifier of restart file The full file name is the identifier extended by a date / time group.
			…/ Part 2

Table 4: WAM_NAMELIST (part 2)

[1]C(l) = CHARACTER (LEN=l); R = REAL; I = INTEGER; L = LOGICAL,
R(n) and I(n) arrays of dimension n, C(l, n) are CHARACTER arrays of (LEN=l) and dimension n
[2] –user- = parameter must be defined by user; -999 = undefined.

Variable name	Type[1]	Default[2]	Description
		Coarse grid	
COARSE_GRID_RUN	L	.FALSE.	Coarse grid option: True: boundary values are saved for a follow-up fine grid run; False: no output
COARSE_OUTPUT_TIMESTEP	I	0	Time step to write boundary spectra for a follow-up fine grid run > 0: output in regular given time steps; <= 0: output every propagation time step
COARSE_OUTPUT_TIMESTEP_UNIT	C(1)	'S'	Unit of time step to write boundary spectra (M = minute, H = hour, S = second)
COARSE_FILE_SAVE_TIMESTEP	I	0	Time step to save coarse grid output boundary files > 0: save in regular given time steps; <= 0: save every output file save time step
COARSE_FILE_SAVE_TIMESTEP_UNIT	C(1)	'S'	Unit of time step to save boundary files (M = minute, H = hour, S = second)
COARSE_OUTPUT_FILE_UNIT	I	70	Logical file unit number to write coarse grid boundary output files. If more than one nest is processed the units are unit number + nest number-1
COARSE_OUTPUT_FILE_NAME	C(80)	'CBO'	File identifier of coarse grid boundary output files. The full file name is the identifier extended by a date / time group. If more than one nest is processed the second and third character are replaced by the nest number.
		Fine grid	

FINE_GRID_RUN	L	.FALSE.	Fine grid option: True: boundary values from a previous coarse grid run are used; False: no influx of energy at grid boundaries
FINE_INPUT_FILE_UNIT	I	2	Logical file unit number to read coarse grid boundary output files
FINE_INPUT_FILE_NAME	C(80)	'CBO'	File identifier of the coarse grid boundary output files to be used for this fine grid run. The full file name is the identifier extended by a date / time group.
Wind input			
WIND_INPUT_TIMESTEP	I	-user-	Time step to use winds from the wind input file. WIND_INPUT_TIMESTEP must be an integer multiple of WIND_OUTPUT_TIMESTEP
WIND_INPUT_TIMESTEP_UNIT	C(1)	'S'	Unit of wind input time step (M = minute, H = hour, S = second)
WIND_OUTPUT_TIMESTEP	I	0	Time step to pass winds to the wave integration. <= 0: wind output time step = wind input time step If (wind output time step < wind input time step) winds are linearly interpolated in time. Wind input time step must be a multiple integer of wind output time step
WIND_OUTPUT_TIMESTEP_UNIT	C(1)	'S'	Unit of wind output time step (M = minute, H = hour, S = second)
WIND_INPUT_FILE_UNIT	I	1	Logical file unit number of wind data input.
WIND_INPUT_FILE_NAME	C(80)	-user-	File name of wind data input file.
		Depth data input	
TOPO_INPUT_TIMESTEP	I	0	Time step to use depth from the depth input file <= 0: depth is stationary
TOPO_INPUT_TIMESTEP_UNIT	C(1)	'S'	Unit of depth input time step (M = minute, H = hour, S = second)
TOPO_OUTPUT_TIMESTEP	I	0	Time step to pass depth data to the wave integration. <= 0: depth output time step = depth input time step If (depth output time step < depth input time step) depth is linearly interpolated in time. Depth input time step must be a multiple integer of depth output time step
TOPO_OUTPUT_TIMESTEP_UNIT	C(1)	'S'	Unit of depth output time step (M = minute, H = hour, S = second)
TOPO_INPUT_FILE_UNIT	I	8	Logical file unit number of depth data input
TOPO_INPUT_FILE_NAME	C(80)	'blank'	File name of depth data input file. If the file does not exist, the depth data are used from PREPROC or out of the restart file.
			…/ Part 3

Table 5: WAM_NAMELIST (part 3)

[1]C(l) = CHARACTER (LEN=l); R = REAL; I = INTEGER; L = LOGICAL,
R(n) and I(n) arrays of dimension n, C(l, n) are CHARACTER arrays of (LEN=l) and dimension n
[2] –user- = parameter must be defined by user; -999 = undefined.

Variable name	Type[1]	Default[2]	Description
		Current data input	
CURRENT_INPUT_TIMESTEP	I	0	Time step to use currents from the current input file <= 0: currents are stationary
CURRENT_INPUT_TIMESTEP_UNIT	C(1)	'S'	Unit of current input time step (M = minute, H = hour, S = second)
CURRENT_OUTPUT_TIMESTEP	I	0	Time step to pass current data to the wave integration. <= 0: current output time step = current input time step If (current output time step < current input time step) currents are linearly interpolated in time. Current input time step must be a multiple integer of current output time step
CURRENT_OUTPUT_TIMESTEP_UNIT	C(1)	'S'	Unit of current output time step (M = minute, H = hour, S = second)
CURRENT_INPUT_FILE_UNIT	I	9	Logical file unit number of current data input
CURRENT_INPUT_FILE_NAME	C(80)	'blank'	File name of current data input file. Only used if current refraction is active. If the file does not exist and current refraction is active, the current data are used out of the restart file.
		Ice data input	
ICE_INPUT_TIMESTEP	I	0	Time step to use ice from the current input file <= 0: ice is stationary
ICE_INPUT_TIMESTEP_UNIT	C(1)	'S'	Unit of ice input time step (M = minute, H = hour, S = second)
ICE_INPUT_FILE_UNIT	I	3	Logical file unit number of ice data input
ICE_INPUT_FILE_NAME	C(80)	'blank'	File name of ice data input file. If file name is "blank" (default) ice is not used in the model run. If the file does not exist, model runs without ice.
		Wave output	
PARAMETER_OUTPUT_TIMESTEP	I	1	Time step to write out integrated parameters

PARAMETER_OUTPUT_TIMESTEP_UNIT	C(1)	'H'	Unit of integrated parameter output time step (M = minute, H = hour, S = second)
PARAMETER_OUTPUT_FILE_UNIT	I	20	Logical file unit number of integrated parameter output
PARAMETER_OUTPUT_FILE_NAME	C(80)	'MAP'	File identifier of integrated parameter output files. The full file name is the identifier extended by a date / time group.
SPECTRA_OUTPUT_TIMESTEP	I	1	Time step to write out spectra
SPECTRA_OUTPUT_TIMESTEP_UNIT	C(1)	'H'	Unit of spectra output time step (M = minute, H = hour, S = second)
SPECTRA_OUTPUT_FILE_UNIT	I	25	Logical file unit number of spectra output
SPECTRA_OUTPUT_FILE_NAME	C(80)	'OUT'	File identifier of spectra output files. The full file name is the identifier extended by a date / time group
OUTPUT_FILE_SAVE_TIMESTEP	I	24	Time step to integrated parameter and spectra output files > 0: save in regular given time steps; <= 0: save at end of run
OUTPUT_FILE_SAVE_TIMESTEP_UNIT	C(1)	'H'	Unit of output file save time step (M = minute, H = hour, S = second)
COUTT	C(14)	'blank'	Up to 20 output date / time groups for integrated parameters and spectra (YYYYMMDDhhmmss) If any date is specified all given output time steps for integrated parameters and spectra will be ignored.
FFLAG_P	L(32)	.TRUE.	File output flag for each integrated parameter type. (Annex E)
PFLAG_P	L(32)	.FALSE.	Printer output flag for each integrated parameter type. (Annex E)
FFLAG_S	L(4)	.TRUE.	File output flag for each spectrum type. (Annex E)
PFLAG_S	L(4)	.FALSE.	Printer output flag for each spectrum type. (Annex E)
OUTLAT	C(13,20)	-999.	Up to 20 latitudes to do spectra output. If not specified spectra output is not done.
OUTLONG	C(13,20)	-999.	Up to 20 longitudes to do spectra output. If not specified spectra output is not done.
NAME	C(20,20)	' '	Optional name given to spectra output sites.
			…/ Part 4

Table 6: WAM_NAMELIST (part 4)

[1]C(l) = CHARACTER (LEN=l); R = REAL; I = INTEGER; L = LOGICAL,
R(n) and I(n) arrays of dimension n, C(l, n) are CHARACTER arrays of (LEN=l) and dimension n
[2] –user- = parameter must be defined by user; -999 = undefined.

Variable name	Type[1]	Default[2]	Description
		Radiation stress output	
RADIATION_OUTPUT_TIMESTEP	I	0	Time step to write out radiation stresses <= 0: output every propagation time step
RADIATION_OUTPUT_TIMESTEP_UNIT	C(1)	'S'	Unit of radiation output time step (M = minute, H = hour, S = second)
RADIATION_FILE_TIMESTEP	I	0	Time step to radiation stress output files > 0: save in regular given time steps; <= 0: save every output file save time step
RADIATION_FILE_TIMESTEP_UNIT	C(1)	'S'	Unit of radiation output file save time step (M = minute, H = hour, S = second)
RADIATION_OUTPUT_FILE_UNIT	I	30	Logical file unit number of radiation stress output
RADIATION_OUTPUT_FILE_NAME	C(80)	'RAD'	File identifier of radiation stress output files. The full file name is the identifier extended by a date / time group
FFLAG_R	L(8)	:TRUE.	File output flag for each radiation stress parameter (Annex F)
PFLAG_R	L(8)	.FALSE.	Printer output flag for each radiation stress parameter (Annex F)
		Preproc output file	
PREPROC_OUTPUT_FILE_UNIT	I	7	Logical file unit number of preproc output
PREPROC_OUTPUT_FILE_NAME	C(80)	Grid_info	Preproc output file name
		Data assimilation	
assimilation_flag	I	0	1 for assimilation; otherwise no assimilation
influence_radius	R	3.0	Radius of influence in degree
observation_scatter	R	0.5	Scatter of observation data
model_scatter	R	0.5	Scatter of model data
assimilation_start_date	C(14)	'blank'	Start date / time group of assimilation (YYYYMMDDhhmmss)
assimilation_end_date	C(14)	'blank'	End date / time group of assimilation (YYYYMMDDhhmmss)
assimilation_time_step	I	3	Assimilation time step
assimilation_time_step_UNIT	C(1)	'H'	Unit of assimilation time step (M = minute, H = hour, S = second)
observation_file_unit	I	80	Logical file unit number of observation input.
observation_filename	C(80)	OBS	File identifier of observation input files. The full file name is the identifier extended by a date / time group
first_guess_output_flag	L	.FALSE.	
first_guess_ip_file_unit	I	30	Logical file unit number of first guess integrated

			parameter output
first_guess_ip_filename	C(80)	MAPFG	File identifier of first guess integrated parameter output files. The full file name is the identifier extended by a date / time group
first_guess_sp_file_unit	I	35	Logical file unit number of first guess spectra output
first_guess_sp_filename	C(80)	OUTFG	File identifier of first guess spectra output files. The full file name is the identifier extended by a date / time group
Special DWD options			
spectral_code	I	-1	If 1 then ASCII output of 2d spectra; otherwise binary output
hours_2d_spectra	I	-1	2d spectra output at all sea points for 'hours_2d_spectra' hours
ready_file_flag	L	.FALSE.	If .True. than the program waits for the next wind field file
ready_file_directory	C(128)	'wind'	File name of ready file.
model_area	C(3)	'GSM'	DWD model area name

VI.5 Post-processing Control Parameters

The post-processing programs PRINT_GRID_FILE, PRINT_SPECTRA_FILE, PRINT_RADIATION_FILE, and PRINT_TIME use the same NAMELIST. In the following xxx denotes GRID, SPECTRA, RADIATION, or TIME.

The code for the input of the control parameter for the post-processing program is in WAM_PRINT_USER_MODULE and the subroutines READ_xxx_USER.

Each program tries to open a file with the name

'Xxx_User'

If the file exists it will try to read the PRINT_NAMELIST. If the reading was not successful, the program will read the same file in a formatted style as required by the program READ_xxx_USER. If the file does not exist the PRINT_NAMELIST is read from standard input.

The NAMELIST variables are listed in Tab. 7. The user has to define the variables marked by "-user-" to assure a successful run.

Default values for the control parameter are defined in subroutine CLEAR_PRINT_USER_MODULE in the WAM_PRINT_USER_MODULE.

If more than 20 output dates or sites should be processed, the parameters `NOUTT` or `MOUTP` in the WAM_PRINT_USER_MODULE must be increased.

All coordinates are CHARACTER (LEN=13) variables. The input can be either in REAL degrees, formatted as F13.8, or formatted as 'sDDD:MM:SS.SS',

where

s is the sign of the coordinate (+ for East and North or – for West and South),
DDD are the degrees,
MM are the minutes and

SS.SS are the seconds of the coordinate (smallest value allowed is 00.02).

Table 7: PRINT_NAMELIST

[1] C(l) = CHARACTER (LEN=l); R = REAL; I = INTEGER; L = LOGICAL,
R(n) and I(n) arrays of dimension n, C(l, n) are CHARACTER arrays of (LEN=l) and dimension n
[2] –user- = parameter must be defined by user; -999 = undefined.

Variable name	Type[1]	Default[2]	Description
		Output times specifications	
START_DATE	C(14)	-user-	Start date / time group of model run (YYYYMMDDhhmmss)
END_DATE	C(14)	-user-	End date / time group of model run (YYYYMMDDhhmmss)
OUTPUT_TIMESTEP	I	1	Time step to write output
OUTPUT_TIMESTEP_UNIT	C(1)	'H'	Unit of output time step (M = minute, H = hour, S = second)
COUTT	C(14,20)	'blank'	Up to 20 output date / time groups (YYYYMMDDhhmmss) If any date is specified an output time step is ignored.
		Data input files	
INPUT_FILE_UNIT	I	20	Logical file unit number of data input.
INPUT_FILE_NAME	C(80)	-user-	File identifier of input file. The full file name is the identifier extended by a date / time group
INPUT_FILE_DATE	C(14)	-user-	Date / time group of first input file (YYYYMMDDhhmmss)
INPUT_FILE_TIMESTEP	I	24	Time step of input files.
INPUT_FILE_TIMESTEP_UNIT	C(1)	'H'	Unit input file time step. (M = minute, H = hour, S = second)
		Output parameter or spectra type selection	
CFLAG_P	L(32)	.TRUE.	File output flag for each integrated parameter type. (Annex E)
CFLAG_S	L(4)	.TRUE.	File output flag for each spectra type. (Annex E)
CFLAG_R	L(8)	.TRUE.	File output flag for each radiation parameter type. (Annex F)
		Output sites for spectra or time series	
OUTLAT	C(13,20)	'blank'	Up to 20 latitudes to do spectra or time series output. If not specified spectra or time series output is not done.
OUTLONG	C(13,20)	'blank'	Up to 20 longitudes to do spectra or time series output. If not specified spectra or time series output is not done.
NAME	C(20,20)	'blank'	Optional name given to spectra or time series output sites.

VII ANNEX B – DATA INPUT

VII.1 Introduction

This annex describes input from various data files, which are necessary for or optional requested by the WAM-Model.
Necessary data input files for a model run are:

- Depth data for the basic model grid for the PREPROC program,
- Wind data for the main WAM model program.

Optionally, controlled by the parameter settings in the *WAM_User* file, the main WAM program requests:

- Sea ice data,
- Depth data,
- Current data,
- Boundary Spectra for a fine grid model run.

For each of these data inputs example programs are provided with the code, which are set-up to read the data files that are included in the data folder. The user may modify the subroutines named READ_xxxx_INPUT, where xxxx denotes the different data sets, for his own file formats.
The input data are transferred from the READ_xxxx_INPUT subroutines into modules by SET subroutines, which are described at the end of this Annex.
Definitions, set-up requirements and algorithms are presented for all data sets in this annex, except the boundary data, which are described in Annex C_Nest.

VII.2 Basic Model Grid and Depth Data

This chapter describes the set-up of the basic model grid for the WAM-Model as done by the PREPROC program.

VII.2.1 Concept of Basic Model Grid

The basic model grid must be a regular latitude/longitude grid, which can have different increments in latitude and longitude. The grid can be but need not East-West periodic. The grid can be defined in the *PREPROC_User* file. If these definitions are not given in the *PREPROC_User* file, the program will use the grid definition of the topographic data input file.
A file containing topographic data on a regular latitude/longitude grid must be assigned to the PREPROC program. The file name has to be defined in the *PREPROC_User* file. The grid need not be identical to the model grid, but must cover the model grid area. The input topographic data are mapped to the basic model grid by using the nearest neighbour method.
Water depth is always positive. Non positive depth greater than a defined cut-up depth less than zero can be included in the basic model grid as dry sea points, which may become wet sea points during the model execution. All grid points with depth less than the defined cut-up depth or land points and not used for further processing.
The depth data can be corrected in up to 20 areas in the basic model grid.
For special model tests a one-point grid can be set-up by the control parameters. This may be used to study duration-limited cases. The model will skip the propagation of wave energy densities.

VII.2.2 Control Parameters

The user must define the filename for a topographic data file. If the model grid is not identical to the topographic in the topographic data file the grid axis have to be defined, too. Optionally up to 20 correction areas, where the water depth should be changed, and the minimum water depth to include land points into the basic model grid can be defined.

See Annex A for details of the control parameters.

VII.2.3 Topographic Data Input

Topographic data are read by the subroutine READ_TOPOGRAPHY. The user may modify the code for own input. The source provided with the code may serve as an example and is set-up to read the topographic file that is included in the data folder.
The subroutine must fulfil the following tasks:

- Open the topographic file with FILE=TRIM(FILE08) and connect it to UNIT=IU08,
- Read the input grid definitions of the topographic field,
- Read the topographic data,
- Call the subroutine SET_TOPOGRAPHY to transfer the data into the PREPROC_MODULE.

To get excess to the subroutines and variables the following USE statements must be inserted:

```
USE PREPROC_MODULE,    ONLY: SET_TOPOGRAPHY
USE WAM_FILE_MODULE,   ONLY: IU06, IU08, FILE08
```

`IU06` is the file unit to write messages into the '*Preproc_Prot*' file,
`IU08` is the file unit of the input topography file as defined in the '*Preproc_User*' file,
`FILE08` is the file name of the input topography file as defined in the '*Preproc_User*' file.

VII.3 Wind Data

This chapter describes the WAM-Model wind data handling. Definitions and set-up requirements are presented.

VII.3.1 Concept of Wind Input

Wind must be on a regular latitude/longitude grid, which need not be identical to the model grid but must cover the model grid area. It can be provided as:

- Components or
- Speed and direction.

The data can be:

- Winds in 10 meters above sea surface (U10),
- Surface stresses (USTRESS), or
- Friction velocities (USTAR).

If the winds are not U10, they are converted to U10 for further processing.

Wind components are first bi-linear interpolated to the model grid. Optionally in a second step the wind speed and wind direction is linearly interpolated in time.

VII.3.2 Wind Control Parameters

The user must define the filename and the wind input time step in the *WAM_User* file. Optionally the wind output time step and the wind file unit can be defined.
The wind input time step is the time increment to use new wind fields from the input file.
The wind output time step is the time step to pass a new wind field to the WAM model. The output time step must be equal to or an integer fraction of wind input time step. In the second case wind fields are linearly interpolated in time. The wind output time step must be an integer multiple of the source function integration time step.

See Annex A for details of the control parameters.

VII.3.3 Wind Data Input

Wind data are read by the subroutine READ_WIND_INPUT, which is called every `WIND_INPUT_TIMESTEP`. The user may modify the code for his wind input. The source provided with the code may serve as an example and is set-up to read the wind file that is included in the data folder.
The subroutine must fulfil the following tasks:

- Open the wind file with FILE=TRIM(FILE01) and connect it to UNIT=IU01,
- Read the input grid definitions of the wind fields and transfer it to the WAM_WIND_MODULE with subroutine SET_WIND_HEADER,
- Read the wind date and wind field and transfer it to the WAM_WIND_MODULE with subroutine SET_WIND_FIELD,
- At each call to READ_WIND_INPUT exactly one date and wind field is read and transferred.

To get excess to the subroutines and variables the following USE statements must be inserted:

```
USE WAM_WIND_MODULE, ONLY: &
&       SET_WIND_FIELD,    & !! WIND INPUT INTO MODULE.
&       SET_WIND_HEADER      !! WIND INPUT HEADER INTO MODULE.

USE WAM_FILE_MODULE, ONLY: IU06, IU01, FILE01
```

`IU06` is the file unit to write messages into the '*WAM_Prot*' file,
`IU01` is the file unit of the input wind file as defined in the '*WAM_User*' file,
`FILE01` is the file name of the input wind file as defined in the '*WAM_User*' file.

VII.4 Sea Ice Data

This chapter describes the WAM-Model sea ice handling. Definitions, set-up requirements and algorithms are presented.

VII.4.1 Concept of Sea Ice Input

The model can optionally be executed

- without ice fields,
- with an ice field constant for model run,
- with ice fields changed at regular time steps during the model run.

Ice maps must be on a regular latitude/longitude grid, which need not be identical to the model grid. It can cover a smaller area than the model grid. The input ice fields are mapped to the model grid using the value at the nearest neighbour in the ice input grid.
Wave spectra at all grid points marked as ice will be set to zero after a propagation step has been done. In the gridded model output ice points are set to '-999'.

VII.4.2 Sea Ice Control Parameters

The user can define the filename and the ice input time step or use the default settings for ice execution.
Sea ice is only taken into account, if a file name is given in the *WAM_User* file. If the file does not exist a warning is printed and the run done without sea ice.
If the ice input time step is not positive, the model will use the first ice field in the file for the full model run.
If the ice input time step is positive ice maps are up-dated during the model run every ice input time step. The model checks before the wave propagation for a new ice map. A new ice map is read and used, if the date of the actually applied ice field plus half of the ice input time step is before the end of the propagation step. The model looks on the file for a new ice map with a date later or equal to the end date of the propagation step.

See Annex A for details of the control parameters.

VII.4.3 Sea Ice Data Input

The subroutine READ_ICE_INPUT may be modified by the user for his ice input. The source provided with the code may serve as an example and is set-up to read the ice file that is included in the data folder.
The subroutine must fulfill the following tasks:

- Open the ice file with FILE=TRIM(FILE03) and connect it to UNIT=IU03,
- Read the input grid definitions of the ice map and transfer it to the WAM_ICE_MODULE with subroutine SET_ICE_HEADER,
- Read the ice date and ice map and transfer it to the WAM_ICE_MODULE with subroutine SET_ICE,
- At each call to READ_ICE_INPUT exactly one ice date and ice map is read and transferred.

To get excess to the subroutines and variables the following USE statements must be inserted:

```
USE WAM_ICE_MODULE,  ONLY: &
&       SET_ICE,           & !! ICE INPUT INTO MODULE.
&       SET_ICE_HEADER       !! ICE INPUT HEADER INTO MODULE.

USE WAM_FILE_MODULE, ONLY: Iu06, IU03, FILE03
```

`IU06` is the file unit to write messages into the '*WAM_Prot*' file,
`IU03` is the file unit of the input ice file as defined in the '*WAM_User*' file,
`FILE03` is the file name of the input ice file as defined in the '*WAM_User*' file.

VII.5 Depth Data

This chapter describes the WAM-Model depth data handling. Definitions and set-up requirements are presented.

VII.5.1 Concept of Depth Data Input

The model can optionally be executed

- without depth fields (the basic depth field is used),
- with a depth field constant for model run,
- with depth fields changed at regular time steps during the model run.

Depth data must be on a regular latitude/longitude grid, which need not be identical to the model grid but must cover the model grid area. They can be provided as:

- Total water depth or
- Surface elevations.

If the depth data are surface elevations, the total water depth used in the WAM-model is surface elevation plus basic water depth as defined by the PREPROC program.
Water depths are first bi-linear interpolated to the model grid. Optionally in a second step the water depths are linearly interpolated in time.

VII.5.2 Depth Control Parameters

The user must define the filename for depth input in the *WAM_User* file. Optionally the depth input and output time step and the depth file unit can be defined.
Depth fields different from the basic model depth are only taken into account, if a file name is given in the *WAM_User* file. If the file does not exist a warning is printed and the run done with the basic model depth.
If the depth input time step is not positive, the model will use the first depth field in the file for the full model run.
If the depth input time step is positive, the depth input time step is the time increment to use a new depth field from the input file. The depth output time step is the time step to pass a new depth field to the WAM model. The output time step must be equal to or an integer fraction of depth input time step. In the second case depth fields are linearly interpolated in time. The depth output time step must be an integer multiple of the propagation integration time step.

See Annex A for details of the control parameters.

VII.5.3 Depth Data Input

Depth data are read by the subroutine READ_TOPO_INPUT, which is called every depth input time step. The user may modify the code for his depth input. The source provided with the code may serve as an example and is set-up to read the depth file that is included in the data folder.
The subroutine must fulfil the following tasks:

- Open the depth file with FILE=TRIM(FILE08) and connect it to UNIT=IU08,

- Read the input grid definitions of the depth fields and transfer it to the WAM_TOPO_MODULE with subroutine SET_TOPO_HEADER,
- Read the depth date and depth field and transfer it to the WAM_TOPO_MODULE with subroutine SET_TOPO_FIELD,
- At each call to READ_TOPO_INPUT exactly one date and depth field is read and transferred.

To get excess to the subroutines and variables the following USE statements must be inserted:

```
USE WAM_TOPO_MODULE, ONLY: &
&       SET_TOPO_FIELD,    & !! DEPTH TO MODULE.
&       SET_TOPO_HEADER      !! DEPTH HEADER TO MODULE.

USE WAM_FILE_MODULE, ONLY: IU06, IU08, FILE08
```

`IU06` is the file unit to write messages into the '*WAM_Prot*' file,
`IU08` is the file unit of the input depth file as defined in the '*WAM_User*' file,
`FILE08` is the file name of the input depth file as defined in the 'WAM_User' file.

VII.6 Current Data

This annex describes the WAM-Model current data handling. Definitions and set-up requirements are presented.

VII.6.1 Concept of Current Input

Current data must be on a regular latitude/longitude grid, which need not be identical to the model grid but must cover the model grid area. They can be provided as:

- Components or
- Speed and direction.

Current components are first bi-linear interpolated to the model grid. Optionally in a second step the current speed and current direction is linearly interpolated in time.

VII.6.2 Current Control Parameters

The user must define the filename for the current input in the *WAM_User* file. Optionally the current input and output time step and the current file unit can be defined.
Currents are only taken into account, if a file name is given in the *WAM_User* file. If the file does not exist a warning is printed and the run done without currents.
If the current input time step is not positive, the model will use the first current field in the file for the full model run.
If the current input time step is positive, the current input time step is the time increment to use a new current field from the input file. The current output time step is the time step to pass a new current field to the WAM model. The output time step must be equal to or an integer fraction of current input time step. In the second case current fields are linearly interpolated in time. The current output time step must be an integer multiple of the propagation integration time step.

See Annex A for details of the control parameters.

VII.6.3 Current Data Input

Current data are read by the subroutine READ_CURRENT_INPUT, which is called every `CURRENT_INPUT_TIMESTEP`. The user may modify the code for his current input. The source provided with the code may serve as an example and is set-up to read the current file that is included in the data folder.

The subroutine must fulfil the following tasks:

- Open the current file with FILE=TRIM(FILE09) and connect it to UNIT=IU09,
- Read the input grid definitions of the current fields and transfer it to the WAM_CURRENT_MODULE with subroutine SET_CURRENT_HEADER,
- Read the current date and current field and transfer it to the WAM_CURRENT_MODULE with subroutine SET_CURRENT_FIELD,
- At each call to READ_CURRENT_INPUT exactly one date and current field is read and transferred.

To get excess to the subroutines and variables the following USE statements must be inserted:

```
USE WAM_CURRENT_MODULE, ONLY: &
&       SET_CURRENT_FIELD,    & !! CURRENT INTO MODULE.
&       SET_CURRENT_HEADER      !! CURRENT HEADER INTO MODULE.

USE WAM_FILE_MODULE, ONLY: IU06, IU09, FILE09
```

`IU06` is the file unit to write messages into the '*WAM_Prot*' file,
`IU09` is the file unit of the input current file as defined in the '*WAM_User*' file,
`FILE09` is the file name of the input current file as defined in the '*WAM_User*' file.

VII.7 Transfer Subroutines

VII.7.1 SET_TOPOGRAPHY Subroutine

Description

Transfers grid definitions and basic topographic input data to the PREPROC_MODULE.

Syntax

```
SET_TOPOGRAPHY (N_LON, N_LAT, D_LON, D_LAT,          &
&               SOUTH, NORTH, WEST, EAST, D_MAP)
```

Required Arguments

N_LON must be of type INTEGER and scalar. It is an INTENT(IN) argument. Its value is the number of longitudes of the topographic data input grid.

N_LAT must be of type INTEGER and scalar. It is an INTENT(IN) argument. Its value is the number of latitudes of the topographic data input grid.

D_LON must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the longitude increment [deg], [s*100] or 'sDDD:MM:SS.SS' of the topographic data input grid.

D_LAT must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the latitude increment [deg], [s*100] or 'sDDD:MM:SS.SS' of the topographic data input grid.

SOUTH must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the southern most latitude [deg], [s*100] or 'sDDD:MM:SS.SS' of the topographic data input grid.

NORTH must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the northern most latitude [deg], [s*100] or 'sDDD:MM:SS.SS' of the topographic data input grid.

WEST must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the western most longitude [deg], [s*100] or 'sDDD:MM:SS.SS' of the topographic data input grid.

EAST must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the eastern most longitude [deg], [s*100] or 'sDDD:MM:SS.SS' of the topographic data input grid.

D_-MAP must be of type INTEGER and an array of rank two. It is an INTENT(IN) argument. Its values are the water depth (positive) and land elevations (negative) [m] of the topographic data. The array must be arranged from WEST to EAST and from SOUTH to NORTH, which is:

```
(    1,      1) <==> South-West corner
(N_LON,      1) <==> South-East corner
(    1, N_LAT) <==> North-West corner
(N_LON, N_LAT) <==> North-East corner
```

Remark:
All coordinates and increments must be of the same type.

VII.7.2 SET_WIND_HEADER Subroutine

Description

Defines the grid of the wind input in WAM_WIND_MODULE.

Syntax

SET_WIND_HEADER (WEST, SOUTH, EAST, NORTH, D_LON, D_LAT, &
& N_LON, N_LAT, CODE)

Required Arguments

WEST must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the western most longitude in [deg], [s*100] or 'sDDD:MM:SS.SS' of the wind input grid.

SOUTH must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the southern most latitude in [deg], [s*100] or 'sDDD:MM:SS.SS' of the wind input grid.

Optional Arguments

EAST must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the eastern most longitude in [deg], [s*100] or 'sDDD:MM:SS.SS' of the wind input grid.

NORTH must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the northern most latitude in [deg], [s*100] or 'sDDD:MM:SS.SS' of the wind input grid.

D_LON must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the longitude increment in [deg], [s*100] or 'sDDD:MM:SS.SS' of the wind input grid.

D_LAT must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the latitude increment in [deg], [s*100] or 'sDDD:MM:SS.SS' of the wind input grid.

N_LON must be of type INTEGER and scalar. It is an INTENT(IN) argument. Its value is the number of longitudes of the wind input grid.

N_LAT must be of type INTEGER and scalar. It is an INTENT(IN) argument. Its value is the number of latitudes of the wind input grid.

CODE must be of type INTEGER and scalar. It is an INTENT(IN) argument. If present, its value is 1 for USTAR, 2 for USTRESS, 3 for U10. If not present input winds are U10.

Remarks:

From the optional arguments two parameters for each grid axis must be provided to assure a complete grid definition. The routine checks the consistency and aborts in case of error.

All coordinates and increments must be of the same type.

VII.7.3 SET_WIND_FIELD Subroutine

Description

Transfers a date, a wind field and a wind code into the WAM_WIND_MODULE.

Syntax

SET_WIND_FIELD (CDT, U_MAP, V_MAP, CODE)

Required Arguments

CDT must be of type CHARACTER (LEN=14) and scalar. It is an INTENT(IN) argument. Its value is the Date/time of the wind field.

U_MAP must be of type REAL and an array of rank two. It is an INTENT(IN) argument. Its values are the u-components or, if CODE is present and equal to one, wind speeds [m/s]. The array must be conformable with the wind grid definition (see subroutine SET_WIND_HEADER). The array must be arranged from WEST to EAST and from SOUTH to NORTH, which is:

```
(    1,      1) <==> South-West corner
(N_LON,      1) <==> South-East corner
(    1, N_LAT) <==> North-West corner
(N_LON, N_LAT) <==> North-East corner
```

V_MAP must be of type REAL and an array of rank two. It is an INTENT(IN) argument. Its values are the v-components [m/s] or, if CODE is present and equal to one, wind directions [deg, coming from]. The array must be organised in the same way as U_MAP.

Optional Arguments

CODE must be of type INTEGER and scalar. It is an INTENT(IN) argument. If present and equal to one U_MAP contains wind speeds and V_MAP wind directions, otherwise arrays contain wind components.

VII.7.4 SET_ICE_HEADER Subroutine

Description

Defines the grid of the ice input map in WAM_ICE_MODULE.

Syntax

SET_ICE_HEADER (WEST, SOUTH, EAST, NORTH, D_LON, D_LAT, N_LON, N_LAT)

Required Arguments

WEST	must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the western most longitude in [deg], [s*100] or 'sDDD:MM:SS.SS' of the ice input grid.
SOUTH	must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the southern most latitude in [deg], [s*100] or 'sDDD:MM:SS.SS' of the ice input grid.

Optional Arguments

EAST	must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the eastern most longitude in [deg], [s*100] or 'sDDD:MM:SS.SS' of the ice input grid.
NORTH	must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the northern most latitude in [deg], [s*100] or 'sDDD:MM:SS.SS' of the ice input grid.
D_LON	must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the longitude increment in [deg], [s*100] or 'sDDD:MM:SS.SS' of the ice input grid.
D_LAT	must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the latitude increment in [deg], [s*100] or 'sDDD:MM:SS.SS' of the ice input grid.
N_LON	must be of type INTEGER and scalar. It is an INTENT(IN) argument. Its value is the number of longitudes of the ice input grid.
N_LAT	must be of type INTEGER and scalar. It is an INTENT(IN) argument. Its value is the number of latitudes of the ice input grid.

Remarks:

From the optional arguments a minimum of parameters for each grid axis must be provided to assure a complete grid definition. The routine checks the consistency and aborts in case of error.

All coordinates and increments must be of the same type.

VII.7.5 SET_ICE Subroutine

Description

Transfers an ice date and an ice map into the WAM_ICE_MODULE and interpolates it to the model grid.

Syntax

SUBROUTINE SET_ICE (CDT, GRID)

Required Arguments

CDT must be of type CHARACTER (LEN=14) and scalar. It is an INTENT(IN) argument. Its value is the date/time of the ice field.

GRID must be of type INTEGER, REAL or CHARACTER (LEN=1) and an array of rank two. It is an INTENT(IN) argument. The array must be conformable with the ice grid definition (see subroutine SET_ICE_HEADER). The array must be arranged from WEST to EAST and from SOUTH to NORTH, which is:

```
(    1,     1) <==> South-West corner
(N_LON,     1) <==> South-East corner
(    1, N_LAT) <==> North-West corner
(N_LON, N_LAT) <==> North-East corner
```

A grid point (i, j) is covered with ice, if GRID (i, j) = 1, NINT(GRID(i,j)) = 1, or GRID(i,j) ==.TRUE., if GRID is of type INTEGER, REAL or CHARACTER, respectively.

VII.7.6 SET_TOPO_HEADER Subroutine

Description

Defines the grid of the depth input in WAM_TOPO_MODULE.

Syntax

```
SET_TOPO_HEADER (WEST, SOUTH, EAST, NORTH, D_LON, D_LAT,         &
&                     N_LON, N_LAT, CODE)
```

Required Arguments

WEST must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the western most longitude in [deg], [s*100] or 'sDDD:MM:SS.SS' of the depth input grid.

SOUTH must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the southern most latitude in [deg], [s*100] or 'sDDD:MM:SS.SS' of the depth input grid.

Optional Arguments

EAST must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the eastern most longitude in [deg], [s*100] or 'sDDD:MM:SS.SS' of the depth input grid.

NORTH must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the northern most latitude in [deg], [s*100] or 'sDDD:MM:SS.SS' of the depth input grid.

D_LON must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the longitude increment in [deg], [s*100] or 'sDDD:MM:SS.SS' of the depth input grid.

D_LAT must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the latitude increment in [deg], [s*100] or 'sDDD:MM:SS.SS' of the depth input grid.

N_LON must be of type INTEGER and scalar. It is an INTENT(IN) argument. Its value is the number of longitudes of the depth input grid.

N_LAT must be of type INTEGER and scalar. It is an INTENT(IN) argument. Its value is the number of latitudes of the depth input grid.

CODE must be of type INTEGER and scalar. It is an INTENT(IN) argument. If present and its value is not equal to 1 input depth are surface elevations. If not present or its value is equal to 1 input depth are total water depth.

Remarks:

From the optional arguments a minimum of parameters for each grid axis must be provided to assure a complete grid definition. The routine checks the consistency and aborts in case of error.

All coordinates and increments must be of the same type.

VII.7.7 SET_TOPO_FIELD Subroutine

Description

Transfers a date and a depth field into the WAM_TOPO_MODULE.

Syntax

SET_TOPO_FIELD (CDT, D_MAP)

Required Arguments

CDT must be of type CHARACTER (LEN=14) and scalar. It is an INTENT(IN) argument. Its value is the date/time of the depth field.

D_MAP must be of type REAL and an array of rank two. It is an INTENT(IN) argument. Its values are the total water depth or surface elevations. The array must be conformable with the depth grid definition (see subroutine SET_TOPO_HEADER). The array must be arranged from WEST to EAST and from SOUTH to NORTH, which is:

```
(     1,      1) <==> South-West corner
(N_LON,      1) <==> South-East corner
(     1, N_LAT) <==> North-West corner
(N_LON, N_LAT) <==> North-East corner
```

VII.7.8 SET_CURRENT_HEADER Subroutine

Description

Defines the grid of the current input in WAM_CURRENT_MODULE.

Syntax

SET_CURRENT_HEADER (WEST, SOUTH, EAST, NORTH, D_LON, D_LAT, &
& N_LON, N_LAT, CODE)

Required Arguments

WEST must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the western most longitude in [deg], [s*100] or ‘sDDD:MM:SS.SS’ of the current input grid.

SOUTH must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the southern most latitude in [deg], [s*100] or 'sDDD:MM:SS.SS' of the current input grid.

Optional Arguments

EAST must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the eastern most longitude in [deg], [s*100] or 'sDDD:MM:SS.SS' of the current input grid.

NORTH must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the northern most latitude in [deg], [s*100] or 'sDDD:MM:SS.SS'] of the current input grid.

D_LON must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the longitude increment in [deg], [s*100] or 'sDDD:MM:SS.SS'of the current input grid.

D_LAT must be of type REAL, INTEGER, or CHARACTER (LEN=13) and scalar. It is an INTENT(IN) argument. Its value is the latitude increment in [deg], [s*100] or 'sDDD:MM:SS.SS' of the current input grid.

N_LON must be of type INTEGER and scalar. It is an INTENT(IN) argument. Its value is the number of longitudes of the current input grid.

N_LAT must be of type INTEGER and scalar. It is an INTENT(IN) argument. Its value is the number of latitudes of the current input grid.

CODE must be of type INTEGER and scalar. It is an INTENT(IN) argument. If present and equal to one U_MAP contains current speeds and V_MAP current directions, otherwise arrays contain current components.

Remarks:

From the optional arguments a minimum of parameters for each grid axis must be provided to assure a complete grid definition. The routine checks the consistency and aborts in case of error.

All coordinates and increments must be of the same type.

VII.7.9 SET_CURRENT_FIELD Subroutine

Description

Transfers a date and a current field into the WAM_CURRENT_MODULE.

Syntax

SET_CURRENT_FIELD (CDT, U_MAP, V_MAP)

Required Arguments

CDT must be of type CHARACTER (LEN=14) and scalar. It is an INTENT(IN) argument. Its value is the date/time of the current field.

U_MAP must be of type REAL and an array of rank two. It is an INTENT(IN) argument. Its values are the u-components or, if CODE is present and equal to one, current speeds [m/s]. The array must be conformable with the current grid definition (see subroutine SET_CURRENT_HEADER). The array must be arranged from WEST to EAST and from SOUTH to NORTH, which is:

```
(    1,     1) <==> South-West corner
(N_LON,     1) <==> South-East corner
```

```
(     1, N_LAT) <==> North-West corner
(N_LON, N_LAT) <==> North-East corner
```

V_MAP must be of type REAL and an array of rank two. It is an INTENT(IN argument. Its values are the v-components [m/s] or, if CODE is present and equal to one, current directions [deg, coming from]. The array must be organised in the same way as U_MAP.

VIII ANNEX C – NEST ORGANISATION AND INTERPOLATION OF SPECTRA

VIII.1 Introduction

This annex describes the WAM-Model nesting strategy. Definitions, set-up requirements and algorithms are presented.

VIII.2 Concept of Nesting

The model can optionally be executed as a

- coarse grid model
- fine grid model
- fine and coarse grid model.

A coarse grid model provides boundary spectra for a follow-up fine grid model, which is embedded as a nest in the coarse grid. A nest is a sub-grid area of the coarse grid area that has a higher grid resolution than the coarse grid. The fine grid model runs on the nest area and uses the boundary spectra provided by the coarse grid model as boundary values. The fine grid model can be a coarse grid model, if a follow-up nest is included in the fine grid model domain.

VIII.3 Nest Set-up in PREPROC Program

The PREPROC program does the organisation of nests. The user has to provide the latitudes and longitudes for the nest boundary to the coarse grid PREPROC and assign the coarse grid PREPROC output file to the fine grid PREPROC program. The necessary set-up routines are collected in WAM_NEST_MODULE.

See Annex A for details of the control parameters.

VIII.3.1 Coarse Grid

A coarse grid model provides boundary spectra for a follow-up fine grid model, which is embedded as a nest in the coarse grid. Therefore the user has to define the fine grid boundaries, which are the West and East latitude and the South and North longitude of the nest, in the '*Preproc_User*' of the coarse grid.
The PREPROC program generates a table that stores all sea point numbers of the nearest sea points along the nest boundaries. The grid points are stored from left to right starting at the lower left corner. In addition to each sea-point number a latitude and longitude is stored in the table. On the West and East boundary of the nest the stored latitude is the coarse grid latitude and the stored longitude is the nearest fine grid longitude. On the South and North boundary of the nest the stored longitude is the coarse grid longitude and the stored latitude is the nearest fine grid latitude. In case that the fine grid corner points are coarse grid points the coarse grid latitudes and longitudes are stored. The maximum shift of coarse grid points is less than half a coarse grid increment in latitude und longitude direction. Land points are ignored.

Figure 18 presents an example nest layout. The table generated by PREPROC in this case is given in Table 8. Table 8 together with the fine grid corner point coordinates are stored in the PREPROC output file and used by the WAM model to write the boundary spectra at the given sea points in the order as given in the table. The fine grid PREPROC uses Table 8 to compute the boundary interpolation table.

The coarse grid PREPROC can generate tables for up to 20 different nested fine grids.

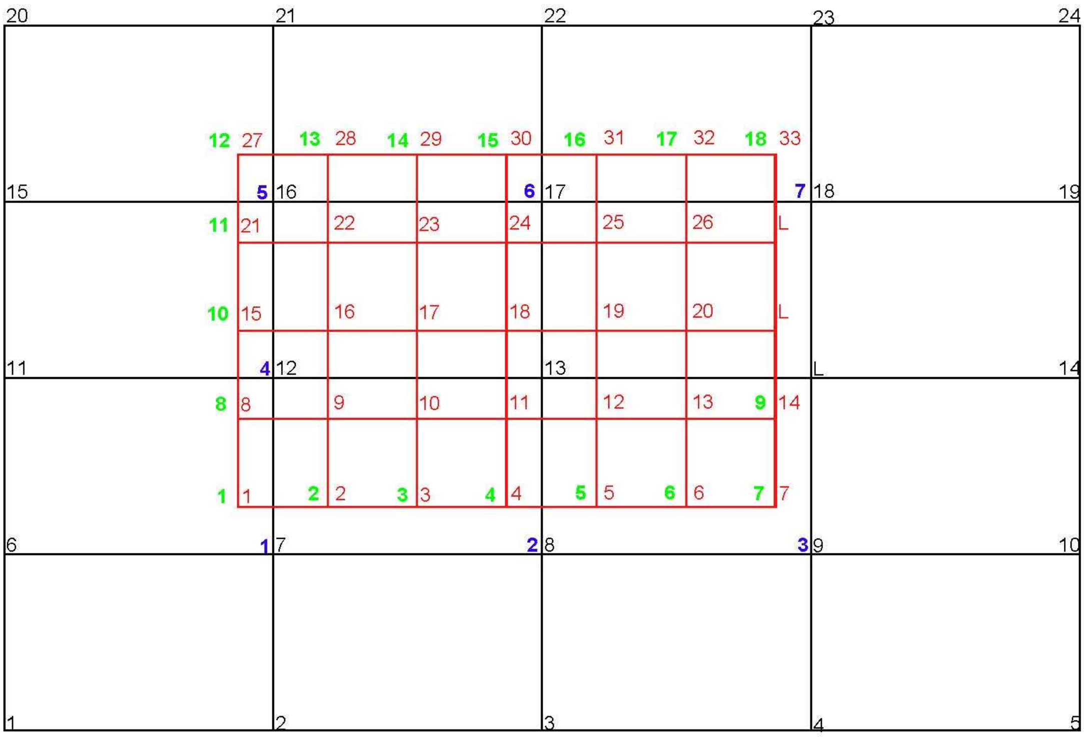

Figure 18: Nest layout

Shown in black are the coarse grid lines and in red are the fine grid lines. The black and red numbers placed to the top right of the grid point are the sea point numbers of the coarse grid and the fine grid, respectively. L marks land points. The blue numbers (placed top-left) count the coarse grid boundary output points and the green numbers count the fine grid input points.

Table 8: Coarse grid output table for the set-up shown in Fig. 18 generated by PREPROC

Table index	Coarse grid sea point number	Assigned Latitude	Assigned Longitude
1	7	Southern fine grid latitude	Western fine grid longitude
2	8	Southern fine grid latitude	Coarse grid longitude at sea point 8
3	9	Southern fine grid latitude	Eastern fine grid longitude
4	12	Coarse grid latitude at sea point 12	Western fine grid longitude
5	16	Northern fine grid latitude	Western fine grid longitude
6	17	Northern fine grid latitude	Coarse grid longitude at sea point 17
7	18	Northern fine grid latitude	Eastern fine grid longitude

VIII.3.2 Fine grid

To set-up a fine grid, the **coarse** grid PREPROC output file has to be assigned to the fine grid PREPROC in the '*Preproc_User*' of the **fine** grid. If more than one coarse grid table is stored the program will identify the table to be used by the stored nest corner points.
The fine grid points at the boundary input points will be numbered counting from left to right starting at the lower left corner (see red numbers in Fig. 18). To each fine grid input point FP the coarse grid table index CP1 of the nearest coarse grid point is assigned. If CP1 and FP have the same latitudes and longitudes CP1 is assigned to FP. Otherwise a second coarse grid table index CP2 is searched, which fulfils the conditions: FP, CP1 and CP2 must have the same latitude or longitude, FP must be between CP1 and CP2 and both coarse grid points are closer than 1.5 coarse grid increments. Coarse grid table indices that do not fulfil the conditions are set to zero.
The spectra at CP1 and CP2 are used for interpolation to FP. Table indices CP1 or CP2, which are zero, are assigned to a spectrum containing zero energy. The interpolation weight stored in the table is the distance of CP1 and FP normalised by the distance of CP1 and CP2. If CP1 is zero the interpolation weight is set to zero. If CP2 is zero, then the distance between CP1 and CP2 is the coarse grid increment. Table 9 displays the fine PREPROC table for the set-up of Fig. 18.

Table 9: Fine grid input table for the set-up shown in Fig. 18 generated by PREPROC

Fine grid boundary input points (green in Fig 1)	Fine grid sea point numbers (red in Fig. 1)	Coarse grid output spectrum 1 (blue in Fig. 1)	Coarse grid output spectrum 2 (blue in Fig. 1)	Interpolation weight
1	1	1	1	1.0000
2	2	1	2	0.2946
3	3	1	2	0.5892
4	4	1	2	0.8838
5	5	2	3	0.2325
6	6	2	3	0.6162
7	7	3	3	1.0000
8	8	1	4	0.6851
9	14	0	3	0.5016
10	15	4	5	0.2127
11	21	4	5	0.6063
12	27	5	5	1.0000
13	28	5	6	0.2946
14	29	5	6	0.5892
15	30	5	6	0.8838
16	31	6	7	0.2325
17	32	6	7	0.6162
18	33	7	7	1.0000

VIII.4 Nest Execution in WAM

The necessary set-up information to run the coarse or fine WAM model is given in the PREPROC output files. In addition the user has to activate the nest execution and define the input or output files in the '*WAM_User*' input file. The necessary execution routines are collected in WAM_BOUNDARY_MODULE and the set-up data are stored in WAM_NEST_MODULE.

VIII.4.1 Coarse Grid

The user must activate the output of boundary values in the '*WAM_User*' input file otherwise nests defined in PREPROC are ignored.
Optionally the user can change the default values for the output time step, the files to save time step, the file name, the file format and the unit number to which the file is assigned. See Annex A User Input for details.
The coarse grid WAM writes the spectra at all sea points given in the coarse grid output table (e.g. Tab. 8) at fixed increments to file. If more than one nest is defined different files for each nest are used. The files are saved and new files are assigned at a given increment.
The output files are in binary or ascii format. The output file contains a file header followed by all spectra in the order as given in the coarse grid output table at same output time. The spectra for the next output time follow immediately.

VIII.4.2 Fine Grid

The user must activate the input of boundary values in the '*WAM_User*' input otherwise nests defined in PREPROC are ignored. The boundary input file name, which is the boundary output file name of the coarse grid WAM, must be given, too.
Optionally the user can change the default value of the unit number to which the input file is assigned. See Annex A User Input for details.
The fine grid WAM reads and stores the boundary spectra for two output times and interpolates the spectra in time to the fine grid model time. The time interpolated spectra are interpolated in space to the boundary input point using the information stored in the fine grid input table generated by fine grid PREPROC (e.g. Tab. 9) and inserted in the model grid. The interpolation of spectra is described in the next chapter.

VIII.5 Interpolation of Spectra

If $F_1(f,\theta)$ and $F_2(f,\theta)$ are spectra at time or location t_1 and t_2, the spectrum $F(f,\theta)$ at time (or location) t is defined by interpolation in 3 steps.

1) The total energies E_1, E_2, mean frequencies $\langle f_1 \rangle$, $\langle f_2 \rangle$ and the mean direction $\langle \theta_1 \rangle$, $\langle \theta_2 \rangle$ for both spectra are computed (see Annex) and linearly interpolated to E, $\langle f \rangle$ and $\langle \theta \rangle$

$$E = E_1 + \frac{t - t_1}{t_2 - t_1}(E_2 - E_1) \tag{1}$$

$$\langle f \rangle = \langle f_1 \rangle + \frac{t - t_1}{t_2 - t_1}\left(\langle f_2 \rangle - \langle f_1 \rangle\right) \tag{2}$$

$$\langle \theta \rangle = \langle \theta_1 \rangle + \frac{t - t_1}{t_2 - t_1}\left(\langle \theta_2 \rangle - \langle \theta_1 \rangle\right) \tag{3}$$

2) The spectra F_1 and F_2 are scaled to have the total energy E, stretched to have the mean frequency $\langle f \rangle$ and rotated to have the mean direction $\langle \theta \rangle$. The resulting spectra G_1 and G_2 have the same integrated parameters E, $\langle f \rangle$ and $\langle \theta \rangle$.

$$G_i(f,\theta) = \frac{E}{E_i} F_i\left(f \frac{\langle f \rangle}{\langle f_i \rangle}, \theta + \langle \theta \rangle - \langle \theta_i \rangle\right) \quad \text{for } i = 1,2 \tag{4}$$

3) The energy densities of G_1 and G_2 at each frequency and direction are linearly interpolated to the spectrum F(f,θ) at time t.

$$F(f,\theta)=G_1(f,\theta)+\frac{t-t_1}{t_2-t_1}\left(G_2(f,\theta)-G_1(f,\theta)\right) \tag{5}$$

VIII.6 Boundary File

The boundary file written by the coarse grid WAM and read by the fine grid WAM can be binary or ascii formatted.
The boundary data are read by the subroutine READ_BOUNDARY_INPUT. The user may modify the code for his input. The source provided with the code may serve as an example and is set-up to read the boundary files that are produced by a standard WAM set-up. If boundary spectra from another source are used, all the information has to be provided as in the standard set-up.

The subroutine must fulfil the following tasks:

- Open the boundary file with FILE=TRIM(FILE02) and connect it to UNIT=IU02,
- Read or define the header information,
- Check consistence with model set-up (recommended),
- Read all boundary spectra for one input time,
- At each call to READ_BOUNDARY_INPUT exactly one set of boundary spectra is read.

To get access to the variables the following USE statements must be inserted:

```
USE WAM_FILE_MODULE,  ONLY: IU06, IU02, FILE02
```

`IU06` is the file unit to write messages into the '*Wam_Prot*' file,
`IU02` is the file unit of the input boundary file as defined in the '*WAM_User*' file,
`FILE02` is the file indicator of the input boundary file as defined in the '*WAM_User*' file.

```
USE WAM_FRE_DIR_MODULE,  ONLY: KL, ML, CO, FR, TH
```

`KL`	is the number of spectral directions as defined in the model set-up,
`ML`	is the number of spectral frequencies as defined in the model set-up,
`CO`	is the logarithmic frequency increment as defined in the model set-up,
`FR(1:ML)`	are the model frequencies as defined in the model set-up,
`TH(1:KL)`	are the model directions as defined in the model set-up.

```
USE WAM_NEST_MODULE,     ONLY: NBINP
```

`NBINP` is the number of input spectra as defined in the nest set-up,

```
USE WAM_BOUNDARY_MODULE, ONLY: CDT_BI_FILE, XLON, XLAT,     &
&                              IDEL_B_INP, IDEL_BI_FILE,    &
&                                         CDATE2, EMEAN2, THQ2,         &   &
FMEAN2, F2
```

`CDT_BI_FILE`	is the date of the presently used boundary file,
`XLON(1:NBINP)`	are the longitudes of the boundary spectra,
`XLAT(1:NBINP)`	are the latitudes of the boundary spectra,
`IDEL_B_INP`	is time step of boundary input spectra,
`IDEL_BI_FILE`	is time step of boundary input spectra file,

`CDATE2`	is date of boundary input spectra to be read,
`EMEAN2(1:NBINP)`	are the total energies of the boundary spectra,
`THQ2(1:NBINP)`	are the mean directions of the boundary spectra,
`FMEAN2(1:NBINP)`	are the mean frequencies of the boundary spectra,
`F2(1:KL,1:ML,1:NBINP)`	are the boundary spectra.

VIII.6.1 Standard Boundary File Format

All values are real numbers and written in binary or ascii (*) format.

1. Record (File Header):
   XANG, XFRE, TH0, FR1, CO1, XBOU, XDELIN, XDELIF

where the values must fulfil the following relations:

```
NINT(XANG)   = KL
NINT(XFRE)   = ML
TH0          = TH(1)
FR1          = FR(1)
CO1          = CO
NINT(XBOU)   = NBINP
NINT(XDELIN)= IDEL_B_INP
NINT(XDELIF)= IDEL_BI_FILE
```

2. Record (Header of first (IJ=1) boundary spectrum):
   XLON(IJ), XLAT(IJ), CDATE2, EMEAN2 (IJ), THQ2(IJ), FMEAN2(IJ)

3. Record (first (IJ=1) boundary spectrum at time CDATE2):
   F2 (1:KL, 1:ML, IJ)

4. until (1 + `2*NBINP`)record:
The 2. and 3. record is repeated for IJ = 2, `NBINP` for all spectra at time `CDATE2`.

Following records contain the boundary spectra at the next time starting with record 2.

IX ANNEX D – MODEL TIME STEPS

IX.1 Introduction

This annex describes time steps in the WAM-Model. Definitions, possibilities and restrictions are presented.

IX.2 Time Steps

The execution of the WAM-Model is controlled by a number of time steps, which are defined by the user.
All time steps are integer values in seconds, which fix the minimum time step to one second.
The basic model time step is the propagation time step. All other time steps with the exception of the source function time step must be an integer multiple of the propagation time step. It is strongly recommended that the propagation and source function steps synchronize soon in time and that output is only done at these times.
To assure a save restart all time steps must synchronize in time at the maximum of the propagation, source function, wind input, depth input, and current input time step.
The model internally does a CFL-check to assure stable integration. If the CFL-criterion is not fulfilled the model internally integrates the advection-refraction with a reduced time step.
The different time steps are cross-checked by the model. If possible the model will correct the steps and warnings are printed in the protocol file otherwise the model will abort and an error message is printed.
All model time steps are listed in the following table:

Table 10: Model time steps

Time step Namelist	Model variable	Purpose	Restriction
PROPAGATION_TIMESTEP	IDELPRO	Propagation integration time step	Propagation time step must fulfill CFL criterion.
SOURCE_TIMESTEP	IDELT	Source function integration time step <= 0: source function = propagation time step	Source function time step should be less than 1200 s in deep and 900 s in shallow water applications.
WIND_INPUT_TIMESTEP	IDELWI	Time step to use winds from the wind input file.	Wind input time step must be a multiple integer of wind output time step. Wind output time step must be a multiple integer of source function time step.
WIND_OUTPUT_TIMESTEP	IDELWO	Time step to pass winds to the wave integration. <= 0: wind output time step = wind input time step. If (wind output time step < wind input time step) winds are linearly interpolated in time.	
TOPO_INPUT_TIMESTEP	IDELTI	Time step to use depth from the depth input file <= 0: depth is stationary	Depth input time step must be a multiple integer of depth output time step. Depth output time step must be a multiple integer of propagation step, if non-stationary depth is used.
TOPO_OUTPUT_TIMESTEP	IDELTO	Time step to pass depth data to the wave integration. <= 0: depth output time step = depth input time step If (depth output time step < depth input time step) depth is linearly interpolated in time.	
CURRENT_INPUT_TIMESTEP	IDELCI	Time step to use currents from the current input	

		file <= 0: currents are stationary	Current input time step must be a multiple integer of depth output time step. Current output time step must be a multiple integer of propagation time step, if non-stationary currents are used.
CURRENT_OUTPUT_TIMESTEP	IDELCO	Time step to pass current data to the wave integration. <= 0: current output time step = current input time step. If (current output time step < current input time step) currents are linearly interpolated in time.	
ICE_INPUT_TIMESTEP	IDEL_ICE_I	Time step to use ice from the ice input file <= 0: ice is stationary	none
PARAMETER_OUTPUT_TIMESTEP	IDELINT	Time step to write out integrated parameters	Output time steps must be a multiple integer of propagation step. File time step must be a multiple integer of the larger output step.
SPECTRA_OUTPUT_TIMESTEP	IDELSPT	Time step to write out spectra	
OUTPUT_FILE_SAVE_TIMESTEP	IDEL_OUT	Time step to integrated parameter and spectra output files > 0: save in regular given time steps; <= 0: save at end of run	
RADIATION_OUTPUT_TIMESTEP	IDEL_RAD_OUT	Time step to write out radiation stresses = 0: output every propagation time step < 0: radiation stresses are not processed.	Output time step must be a multiple integer of propagation step. File time step must be a multiple integer of output time step.
RADIATION_FILE_TIMESTEP	DELFIL	Time step to radiation stress output files > 0: save in regular given time steps; <= 0: save every output file save time step	
RESTART_SAVE_TIMESTEP	IDEL_RES	Time step to save restart files > 0: restart files are saved in regular time steps. = 0: restart file is saved at the end of run. < 0: restart file is not saved.	Restart time step must be a multiple integer of propagation or source time step, whichever is larger.
COARSE_OUTPUT_TIMESTEP	IDEL_B_OUT	Time step to write boundary spectra for a follow-up fine grid run > 0: output in regular given time steps; <= 0: output every propagation time step	Output time step must be a multiple integer of propagation or source time step, whichever is larger. File step must be a multiple integer of output time step.
COARSE_FILE_SAVE_TIMESTEP	IDEL_BO_FILE	Time step to save coarse grid output boundary files > 0: save in regular given time steps; <= 0: save every output file save time step	

X ANNEX E – WAVE OUTPUT

X.1 Introduction

This annex describes the WAM-Model output parameters. Definitions and algorithms to compute the model output from the energy density spectrum, which is the prognostic variable of the WAM-Model, are presented.

X.2 Concept of Spectra and Spectral Parameter

X.2.1 Spectra

The surface variance spectrum is proportional to the wave energy density spectrum. Therefore the term energy density spectrum is mostly used in the wave modelling community.
The equation solved is the mathematical description of physical conservation law of action density. The energy density is computed at all grid points for all time steps. Neglecting in the following the space and time dependence the energy density $F_i(f_i,\theta)$ is represented in the model as function of intrinsic frequency f_i (dimension 1/s) and wave direction θ (dimension radiance). The dimension of the energy density $F_i\,(f_i,\theta)$ is m^2/Hz/rad.
The intrinsic frequency and the absolute frequency f are connected via the Doppler term

$$2\pi f = 2\pi f_i - \mathbf{ku} \tag{1}$$

where **u** is the current vector and **k** is the wave number. The modulus k of **k** is defined by the dispersion relation

$$2\pi f_i = \sigma = \sqrt{gk\tanh(kd)} \tag{2}$$

where d denotes the water depth.

To get the 2-D output spectrum $F\,(f,\theta)$ based on the absolute frequency the transformation

$$F(f,\theta) = F_i(f_i,\theta)\left|\frac{\partial f_i}{\partial f}\right| \tag{3}$$

is performed. If the model runs without currents, the intrinsic frequency f_i and the absolute frequency f are identical. Transformations of the spectrum are not necessary, and $F(f,\theta) = F(f_i,\theta)$.

The 2-D spectrum is reduced to the 1-D spectrum $E(f)$ by integration over the directions.

$$E(f) = \int_0^{2\pi} F(f,\theta)d\theta \tag{4}$$

The wave spectrum $E(f)$ is equivalent to wave spectra deduced from time series measurement, e.g. by waverider buoys.

The directional distribution $R(f,\theta)$ is defined by

$$R(f,\theta)=\frac{F_2(f,\theta)}{E(f)} \tag{5}$$

With the definitions

$$\begin{aligned}\overline{S\theta}(f)&=\int_0^{2\pi}\sin\theta F_2(f,\theta)d\theta\\ \overline{C\theta}(f)&=\int_0^{2\pi}\cos\theta F_2(f,\theta)d\theta\end{aligned} \tag{6}$$

the mean direction per frequency (in radiance) is given as

$$\overline{\theta(f)}=\arctan\left(\frac{\overline{S\theta}(f)}{\overline{C\theta}(f)}\right) \tag{7}$$

and the directional spread per frequency (in radiance) as

$$s(f)=\sqrt{2-2\frac{\sqrt{\overline{S\theta}^2(f)+\overline{C\theta}^2(f)}}{E(f)}} \tag{8}$$

X.2.2 Integrated Wave Parameter

In the following a number of integrated parameters, which are frequently used, are defined:

- Spectral moment of order i

$$m_i=\int f^i E(f)df \tag{9}$$

- o Significant wave height [m]

$$H_s=4\sqrt{m_0} \tag{10}$$

- Mean wave period [s]

$$T_{mean}=\frac{m_{-1}}{m_0} \tag{11}$$

- Wave Tm1 period [s]

$$T_{M1}=\frac{m_0}{m_1} \tag{12}$$

- Wave Tm2 period [s]

$$T_{M2} = \sqrt{\frac{m_0}{m_2}} \tag{13}$$

- Wave Peak period [s]

$$T_p = \frac{1}{f_p} \; where \; \mathrm{E}(\mathrm{f_p}) = \underset{0<f}{Max}\{E(f)\} \tag{14}$$

Mean directional wave parameters are based on

$$\begin{aligned} \overline{S\theta} &= \int \overline{S\theta}(f)df \\ \overline{C\theta} &= \int \overline{C\theta}(f)df \end{aligned} \tag{15}$$

- Mean direction [rad]

$$\theta_m = \arctan\left(\frac{\overline{S\theta}}{\overline{C\theta}}\right) \tag{16}$$

- Mean directional spread [rad]

$$\sigma_m = \sqrt{2 - 2\frac{\sqrt{\overline{S\theta}^2 + \overline{C\theta}^2}}{m_0}} \tag{17}$$

X.2.3 Wind Sea and Swell

A total wave spectrum can be separated in a windsea- (or sea-) and swell- spectrum. The sea spectrum is the part of the total spectrum, which is under the influence of the local wind speed. The remaining part of the total spectrum is called swell. The term “under the influence of the local wind speed” means that the phase speed of the wave–components is less than the friction velocity assigned to local wind speed component. If u_{10} denotes the wind speed and θ_{wind} the wind direction in 10m above sea surface, a spectral component $F(f,\theta)$ is defined as swell if

$$\frac{g}{2\pi f} > 1.2\eta u_* \cos(\theta - \theta_{wind}) \tag{18}$$

where the friction factor $\eta = 28$ and u_* is the friction velocity corresponding to u_{10} .

X.3 Algorithmic Implementation

This section describes the technical aspects of the algorithmic implementation that are relevant to the validation process. In particular, it addresses the differences between the algorithmic implementation and the conceptual model.

X.3.1 Spectral Domain

The frequency axis f of the energy density spectra $F(f,\theta)$ is discretized by f_l ,l =1,…,M_f , where the minimum frequency f_1 >0 and the following numbers increase logarithmically by 10%

$$f_l = 1.1^{l-1} f_1 \tag{19}$$

The discrete frequency f_l is the centre of the frequency interval Δf_l with the left and right boarder $f_{a,l-1}$ and $f_{a,l}$, respectively, where

$$\begin{aligned} f_{a,0} &= f_1 - 0.5(f_2 - f_1) \\ f_{a,l} &= 0.5(f_l + f_{l+1}) \quad \text{if} \quad l = 1,\ldots,M_f - 1 \\ f_{a,M_f} &= f_{M_f} + 0.5(f_{M_f} - f_{M_f-1}) \end{aligned} \tag{20}$$

The direction axis θ is discretized by θ_μ ,μ =1,…,M_θ. The axis has to cover the full circle in equidistant steps $\Delta\theta$ and is therefore completely defined by the number of directions M_θ. It is

$$\begin{aligned} &\Delta\theta = 2\pi / M_\theta \\ &\theta_\mu = \left(\mu - \frac{1}{2}\right)\Delta\theta \quad \mu = 1,\ldots,M_\theta \end{aligned} \tag{21}$$

The discrete direction θ_μ is the centre of the interval [θ_μ-0.5$\Delta\theta$, θ_μ+0.5$\Delta\theta$].

X.3.2 Transformation from Intrinsic to Absolute Frequencies

The complete model output is based on absolute frequencies. In case currents are used in the model, spectra are transformed from intrinsic to absolute frequencies based on equ. (3), before any integrated parameters are computed or spectra output is done.

Step 1:
For each discrete intrinsic model frequency f the absolute frequency f_a is computed from the dispersion relation equ. (2) (frequency and directional indices are not show in the following).

Step 2:
The spectral energy densities at the absolute frequencies are computed as

$$\begin{aligned} &F_a(f_a,\theta) = F(f,\theta)\frac{\Delta f}{\Delta f_a} && \text{if } f_a > 0 \\ &F_a(-f_a,\theta+180^0) = F(f,\theta)\frac{\Delta f}{\Delta f_a} && \text{if } f_a < 0 \end{aligned} \tag{22}$$

where the frequency intervals are defined as in equ. (20).

Step 3:
The energy densities of spectra at the absolute frequencies f_a as defined in step 2 are redistributed to the discrete model frequency.

X.3.3 The Output Energy Density Spectral Domain

The model output are 2-D frequency – direction surface variance spectra $F(f_j, \theta_\mu)$ in the absolute frequency coordinates as defined in E.3.2.
A discrete output frequency axis f_j ,$j = 1,\dots,M_f$ and the discrete direction axis is equal to the axis used in the computations of the energy density spectra cf. equ. (19-21).

X.3.4 Computation of Output Integrated Parameter

The integrated parameters are computed from the absolute spectra $F(f_j, \theta_\mu)$. For the computations, based on the definitions equ. (4) - (17), all integrals are replaced by summations, e.g. the 1-D frequency spectrum (cf. equ. (4)) is computed as

$$E\left(f_j\right) = \sum_{\mu=1}^{M_\theta} F\left(f_j, \theta_\mu\right)\Delta\theta \tag{23}$$

In addition a tail correction T_i is added to the moments $i = -1,0,1, and\ 2$ to account for the frequency cut-off at f_{Mf}. It is assumed that the energy densities for frequencies greater than f_{Mf} are proportional f^{n}, where $n = 5$ is fixed in the WAM-Model.

With these assumptions the tail function is given as

$$t(f) = af^{-n} \tag{24}$$

where a is a constant.

Because f_{Mf} is the cut-up frequency and $E(f_{Mf})$ the energy density at the cut-up frequency, the tail function must fulfil

$$t(f_{M_f}) = E\left(f_{M_f}\right) \tag{25}$$

which defines a as

$$a = f_{M_f}^{n} E\left(f_{M_f}\right) \tag{26}$$

For the i-moment the tail contribution is defined by

$$T_i = a\int_{f_{M_f}}^{\infty} f^{i} f^{-n} df \tag{27}$$

$$T_i = \frac{E(f_{M_f}) f_{M_f}^{i+1}}{n - i - 1} \tag{28}$$

With this correction the moment of order i is computed as

$$m_i = T_i + \sum_{j=1}^{M_f} f_j^i E(f_j) \Delta f_j \tag{29}$$

X.3.5 Computation of Output Wind Sea and Swell Parameters and Spectra

Windsea and Swell integrated parameters are computed in the same way as described in E3.1-E3.3, but before doing the calculation, the frequency-direction domain is restricted.

For the windsea integration the sub- domain is used, that fulfils equ. 18 and for the swell integration the sub- domain is used, that does not fulfill equ. 18.

X.4 Output Files

This chapter describes the wave output files of the WAM model.
Controlled by the setting of the parameters in the *WAM_User* file the model generates separate wave output files for integrated parameters and/or wave spectra. Output can be written into the formatted *WAM_Prot* file and/or into automatically assigned binary files. The filenames of the binary files consist of a file identifier extended by a date / time group. These files are opened at fixed time increments or one file is used for the full model run.
Output is written in fixed time increments or at special output times as defined in the *WAM_User* file. The date / time group included in the file name is the date /time of the last output stored in the file.
See Annex A for details of the control parameters.

X.4.1 Integrated Parameter Output File

All output parameters available in the model, which can be selected for output in the *WAM_User* file, are given in Table 11. The parameter fields are gridded and a missing value indicator is written at land point. For the formatted output into the *WAM_Prot* file the parameters are scaled to integer values.
The integrated parameter output files can be read by the subroutine READ_GRID_FILE and further processed by the programs PRINT_GRID_FILE, which prints formatted parameter fields, and PRINT_TIME, which prints time series at selected grid points.
The file format is:

1. Record (Header):

```
CDTINTT, DNX, DNY, AMOWEP, AMOSOP, AMOEAP, AMONOP, cstart
```

where

`CDTINTT`	is the date of output field (YYYYMMDDhhmmss)
`NINT(DNX)` = NX	is the number of grid points in West-East direction
`NINT(DNY)` = NY	is the number of grid points in North-South direction
`AMOWEP`	is the most western latitude of grid
`AMOSOP`	is the most southern longitude of grid

```
AMOEAP          is the most eastern latitude of grid
AMONOP          is the most northern longitude of grid
cstart          is the start date of model run (YYYYMMDDhhmmss)
```

2. Record (Data flag array):

```
PFLAG_P(1:32)
```

where

`PFLAG_P(I)` is true, if parameter field I (see Tab. 11) is included in the file.

3. and following records (one record for each parameter, where `PFLAG_P(I)=.TRUE.`):

```
GRID(1:NX,1:NY)
```

where

GRID is the gridded field of parameter I. The array is arranged from WEST to EAST and from NORTH to SOUTH, which is:

```
(    1,     1) <==> North-West corner
(N_LON,     1) <==> North-East corner
(    1, N_LAT) <==> South-West corner
(N_LON, N_LAT) <==> South-East corner
```

Following records contain the output data at the next time starting with record 1.

Table 11: Integrated output parameter

Parameter No.	Parameter	Dimension
1	Wind speed U10	m/s
2	Wind direction	Degree from North (towards)
3	Friction velocity	m/s
4	Drag coefficient	
5	Water depth	m
6	Current speed	m/s
7	Current direction	Degree from North (towards)
8	Dummy	
9	Significant wave height	m
10	Wave peak period	s
11	Wave mean period	s
12	Wave Tm1 period	s
13	Wave Tm2 period	s
14	Wave direction	Degree from North (towards)
15	Directional spread	Degree
16	Normalized wave stress	%
17	Sea significant wave height	m
18	Sea peak period	s
19	Sea mean period	s
20	Sea Tm1 period	s
21	Sea Tm2 period	s
22	Sea direction	Degree from North (towards)
23	Sea directional spread	Degree
24	Dummy	
25	Swell significant wave height	m
26	Swell peak period	s
27	Swell mean period	s
28	Swell Tm1 period	s
29	Swell Tm2 period	s
30	Swell direction	Degree from North (towards)
31	Swell directional spread	Degree
32	Dummy	

X.4.2 Spectra Output File

All output spectra types available in the model, which can be selected for output in the *WAM_User* file, are given in Table 12. The spectra are written at selected sea points, which are defined in the *WAM_User* file. Spectra are written to binary output files and/or into the formatted *WAM_Prot* file.
The spectra output files can be read by the subroutine READ_SPECTRA_FILE and further processed by the program PRINT_SPECTRA_FILE, which prints formatted spectra.
The files format is:

1. Record (Header):

```
SPEC_LON, SPEC_LAT, SPEC_DATE, XANG, XFRE, TH1, FR1, CO
```

where

`SPEC_LON`	is the longitude of the spectra
`SPEC_LAT`	is the latitude of the spectra
`SPEC_DATE`	is the date of the spectra (YYYYMMDDhhmmss)
`NINT(XANG) = KL`	is the number of spectral directions
`NINT(XFRE) = ML`	is the number of spectra frequencies
`TH1`	is the first spectral direction
`FR1`	is the first spectral frequency
`CO`	is the logarithmic frequency increment

2. Record (Data flag array):

```
PFLAG_S(1:4)
```

where
`PFLAG_S(I)` is true, if spectra type I (see Tab. 12) is included in the file.

3. Record (Environment parameter):

```
U10, UDIR, US, DEPTH, CSPEED, CDIR
```

where

`U10`	is the wind speed u10
`UDIR`	is the wind direction in degree from North (towards)
`US`	is the friction velocity
`DEPTH`	is the water depth
`CSPEED`	is the current speed
`CDIR`	is the current direction in degree from North (towards)

4. Record (wave parameters for the first spectrum type I where `PFLAG_S(I)=.TRUE.`):

```
HS, PPER, MPER, TM1, TM2, MDIR, SPRE
```

where

`HS`	is the significant wave height
`PPER`	is the peak period
`MPER`	is the mean period
`TM1`	is the Tm1 period
`TM2`	is the Tm2 period
`MDIR`	is the wave direction in degree from North (towards)
`SPRE`	is the directional spread in degree

5. Record (wave spectrum for the first spectrum type I where `PFLAG_S(I)=.TRUE.`):

```
SPEC(1:KL,1:ML)
```

Records 5 and 6 are repeated for the other spectra types I, if `PFLAG_S(I)=.TRUE.`

The following records contain the output spectra at the same time but at the next output sites always starting with record 1.

After that the record sequence is repeated for the next output time.

Table 12: Spectra output types

Spectra type No.	Spectra type	Dimension
1	Wave spectrum	m*m/(Hz*rad)
2	Sea spectrum	m*m/(Hz*rad)
3	Swell spectrum	m*m/(Hz*rad)
4	Dummy	

XI ANNEX F – RADIATION STRESS, WAVE FORCE AND STOKES DRIFT OUTPUT

XI.1 Introduction

This annex describes the WAM-Model radiation stress, wave force and stokes drift computation and output. Definitions and algorithms to compute the output from the energy density spectrum, which is the prognostic variable of the WAM-Model, are presented.

XI.2 Definitions

XI.2.1 Radiation Stress Tensor

The radiation stress tensor S = S_{ij} is defined as

$$S_{ij} = \rho_w \, g \int_0^{2\pi} \int_0^{\infty} \left[\frac{c_g}{c} \frac{k_i k_j}{k^2} + \left(\frac{c_g}{c} - \frac{1}{2} \right) \delta_{ij} \right] F(f,\theta) \, df \, d\theta \tag{1}$$

where

$F(f,\theta)$ is the energy density spectrum at intrinsic frequency f and direction θ, c_g the group velocity, c the phase velocity, g the acceleration of gravity, ρ_w the water density. The wave number components k_i, k_j and the wave number modulus k are functions of frequency and direction. The indices i and j denote the components in x (West-East) and y (South-North) direction, respectively. The terms k_i/k and k_j/k are given as $\sin\theta$ and $\cos\theta$, respectively.

XI.2.2 Wave Force per Surface Unit

The wave force per surface unit vector $\boldsymbol{\tau} = (\tau_x, \tau_y)$ used is defined as

$$\tau_x = -\frac{1}{\rho_w} \left(\frac{\partial S_{xx}}{\partial x} + \frac{\partial S_{xy}}{\partial y} \right) \tag{2}$$

$$\tau_y = -\frac{1}{\rho_w} \left(\frac{\partial S_{yx}}{\partial x} + \frac{\partial S_{yy}}{\partial y} \right) \tag{3}$$

where S_{ij} are the radiation stress tensor elements.

XI.2.3 Stokes Drift

From a directional wave spectrum F(f,θ), where , f denotes the intrinsic frequency and θ the wave direction, the Stokes drift vector $\mathbf{u_s}$ = (u, v**)** is defined as:

$$\mathbf{u}_s = \frac{4\pi}{g}\int_0^{2\pi}\int_0^{\infty} f\mathbf{k}F(f,\theta)dfd\theta \tag{4}$$

where g denotes acceleration of gravity and **k** the wave number vector.

XI.3 Computations

This section describes the technical aspects of the algorithmic implementation that are relevant to the validation process. In particular, it addresses the differences between the algorithmic implementation and the conceptual model.

XI.3.1 Radiation Stress Tensor Elements

The discrete frequencies f_l *(l=1,…, M_l)* and direction θ_μ *(μ=1,…, M_μ)* axis are given in Annex E section E3.1. The radiation stress tensor elements at each sea point are computed based on equ. (1). The integral is replaced by summations.

$$\begin{aligned} S_{xx} &= \rho_w g \sum_{l=1}^{M_l} \frac{c_g(f_l)}{c(f_l)} \sum_{\mu=1}^{M_\theta} \sin^2\theta_\mu F(f_l,\theta_\mu)\Delta\theta_\mu \Delta f_l \\ &+ \rho_w g \sum_{l=1}^{M_l} \left[\frac{c_g(f_l)}{c(f_l)} - \frac{1}{2}\right] \sum_{\mu=1}^{M_\theta} F(f_l,\theta_\mu)\Delta\theta_\mu \Delta f_l \end{aligned} \tag{5}$$

$$\begin{aligned} S_{yy} &= \rho_w g \sum_{l=1}^{M_l} \frac{c_g(f_l)}{c(f_l)} \sum_{\mu=1}^{M_\theta} \cos^2\theta_\mu F(f_l,\theta_\mu)\Delta\theta_\mu \Delta f_l \\ &+ \rho_w g \sum_{l=1}^{M_l} \left[\frac{c_g(f_l)}{c(f_l)} - \frac{1}{2}\right] \sum_{\mu=1}^{M_\theta} F(f_l,\theta_\mu)\Delta\theta_\mu \Delta f_l \end{aligned} \tag{6}$$

$$S_{xy} = S_{yx} = \rho_w g \sum_{l=1}^{M_l} \frac{c_g(f_l)}{c(f_l)} \sum_{\mu=1}^{M_\theta} \sin\theta_\mu \cos\theta_\mu F(f_l,\theta_\mu)\Delta\theta_\mu \Delta f_l \tag{7}$$

XI.3.2 Wave Force per Surface Unit

The wave force per surface unit vector elements are computed from equ. (2) and (3) at each sea point. The gradients in the equations are computed as centred finite differences. If one of the neighbour grid points is a land, sea or dry point, first order finite differences are used. (In case of a nested fine grid all boundary input points are treated as land points.) If both neighbours do not exist, the radiation stress vector elements are set to undefined.

XI.3.3 Stokes Drift

The discrete frequencies f_l *(l=1,…, M_l)* and direction θ_μ *(μ=1,…, M_μ)* axis are given in Annex E section E3.1. The Stokes drift vector elements at each sea point are computed based on equ. (4). The integral is replaced by summations.

$$\mathbf{u}_s = \frac{4\pi}{g}\sum_{l=1}^{M_l}\sum_{\mu=1}^{M_\mu} f_l \mathbf{k}(f_l) F(f_l, \theta_\mu) \Delta f_l \Delta\theta_\mu \tag{8}$$

In addition a tail correction T_3 is added to the third moments to account for the frequency cut-off at f_{Mf}. It is assumed that the energy densities for frequencies greater than f_{Mf} are proportional f^n, where $n = 5$ is fixed in the WAM-Model (cf. Annex E section E3.4).

XI.4 Output File

Controlled by the setting of the parameters in the *WAM_User* file the model generates output of these parameters. Output can be written into the formatted *WAM_Prot* file and/or into automatically assigned binary files. The filenames of the binary files consist of a file identifier extended by a date / time group. These files are opened at fixed time increments or one file is used for the full model run.
Output is written in fixed time increments as defined in the *WAM_User* file. The date / time group included in the file name is the date /time of the last output stored in the file.
All output parameters available in the model, which can be selected for output in the *WAM_User* file, are given in Table 13. The parameter fields are gridded and a missing value indicator is written at land, ice and dry points. For the formatted output into the *WAM_Prot* file the parameters are scaled to integer values.
See Annex A for details of the control parameters.
The binary output files can be read by the subroutine READ_RADIATION_FILE and further processed by the programs PRINT_RADIATION_FILE, which prints formatted parameter fields.

The file format is:

1. Record (Header):

```
CDTINTT, DNX, DNY, AMOWEP, AMOSOP, AMOEAP, AMONOP
```

where

`CDTINTT`	is the date of output field (YYYYMMDDhhmmss)
`NINT(DNX)` = NX	is the number of grid points in West-East direction
`NINT(DNY)` = NY	is the number of grid points in North-South direction
`AMOWEP`	is the most western latitude of grid
`AMOSOP`	is the most southern longitude of grid
`AMOEAP`	is the most eastern latitude of grid
`AMONOP`	is the most northern longitude of grid

2. Record (Data flag array):

```
PFLAG_R(1:8)
```

where
`PFLAG_R(I)` is true, if parameter field I (see Tab. 13) is included in the file.

3. and following records (one record for each parameter, where `PFLAG_R(I)=.TRUE.`):

```
GRID(1:NX,1:NY)
```

where

GRID is the gridded field of parameter I. The array is arranged from WEST to EAST and from NORTH to SOUTH, which is:

```
(    1,     1) <==> North-West corner
(N_LON,     1) <==> North-East corner
(    1, N_LAT) <==> South-West corner
(N_LON, N_LAT) <==> South-East corner
```

Following records contain the output data at the next output time starting with record 1.

Table 13: Radiation stress output parameter

Parameter No.	Parameter	Dimension
1	Radiation Stress Tensor S_{xx}	kg/s^2
2	Radiation Stress Tensor S_{yy}	kg /s^2
3	Radiation Stress Tensor S_{xy}	kg /s^2
4	Dummy	
5	x- comp. Wave Force per Surface Unit	N/m^2
6	y- comp. Wave Force per Surface Unit	N/m^2
7	x- comp. Stokes Drift	m/s^2
8	y- comp. Stokes Drift	m/s^2

XII ANNEX G – REDUCED GRID

XII.1 Introduction

This annex describes the WAM-Model reduced grid set-up. Definitions and algorithms to compute the set-up and consequences for the calculation spatial gradients are presented.

XII.2 Definition of the Reduced Grid

Due to the convergence of longitudes the distance in metres between longitudes is reduced towards the poles. This results in an unbalanced grid resolution and requires a strong reduction of the propagation time step to avoid numerical instability. This can be overcome by reducing the number of grid points on high latitudes.

A regular grid is defined by

(x_0, y_0) the west-south corner coordinate (in degrees),
(d_x, d_y) the increments in west-east and in south-north direction(in degrees),
(N_x, N_y) the number of latitudes and longitudes,
(x_{Nx}, y_{Ny}) the east-north corner coordinate (in degrees)

and the definition must fulfil the relations:

$$\begin{aligned} x_{Nx} &= x_0 + (N_x - 1)d_x \\ y_{Ny} &= y_0 + (N_y - 1)d_y \end{aligned} \tag{1}$$

and the coordinates (x_i, y_k) of all grid point are defined as:

$$\begin{aligned} x_i &= x_0 + (i-1)d_x \quad & i = 1,\ldots,N_x \\ y_k &= y_0 + (k-1)d_y \quad & k = 1,\ldots,N_y \end{aligned} \tag{2}$$

The regular grid is converted into a reduced grid as follows:

If *l* is the latitude, where

$$\cos(y_l) = \underset{k=1}{\overset{Ny}{Max}}\{\cos(y_k)\} \tag{3}$$

reduced number of grid points $N_x(k)$ for each latitude *k* is defined in three steps:

$$
\begin{aligned}
&N_x(k) = NINT\left[N_y \frac{\cos(y_k)}{\cos(y_l)}\right] \\
&\text{if } N_x(k) \text{ is an odd number}: \; N_x(k) = N_x(k) + 1 \\
&N_x(k) = Minimum\left[N_x, N_x(k)\right]
\end{aligned}
\tag{4}
$$

where NINT denotes the nearest integer number.

The longitude increments of the reduced grid for each latitude $d_x(k)$ are consequently define as

$$
\begin{aligned}
&d_x(k) = \frac{x_{Nx} - x_0}{N_x(k) - 1} \quad \text{if the grid is non-periodic in longitude} \\
&d_x(k) = \frac{x_{Nx} + d_x - x_0}{N_x(k)} \quad \text{if the grid is periodic in longitude}
\end{aligned}
\tag{5}
$$

Finally the coordinates (x_i, y_k) of all grid point of the reduced grid are defined as:

$$
\begin{aligned}
&x_i = x_0 + (i-1)d_x(k) \quad i = 1,\ldots,N_x(k) \\
&y_k = y_0 + (k-1)d_y \quad k = 1,\ldots,N_y
\end{aligned}
\tag{6}
$$

With these definitions follows that

- $N_x(l) = N_x$,
- The number of grid points on reduced latitudes is even,
- The grid boundaries are kept the same as in the full grid.
- The longitude increment $d_x(k) = d_x$ for all latitudes, where $N_x(k) = N_x$

Remark:
To use the same code for a reduced and a regular grid. The values $d_x(k)$ and $N_x(k)$ are defined for the regular grid too and fixed to the constants d_x and N_x , respectively.

XII.3 Gradients

Spatial gradients have to be computed for the spectral components, depth und current data.
In the model this is controlled by index arrays, which store the sea point number of the four neighbour grid point in West, East, South and North direction.

East –West gradients
The indices of the neighbour points are calculated in the same way as in the full grid. In the computation of the gradients the reduced grid latitude increment $d_x(k)$ is used.

North –South gradients
The north and south neighbour grid points are selected as nearest to the longitude of the actual grid point. The increment used is the constant latitude increment d_y.

XII.4 Reduced Grid Output

All print and file output of the main program CHIEF is on the reduced grid. The values $d_x(k)$ and $N_x(k)$ are stored in the file output. The post-processing programs PRINT_GRID_FILE and PRINT_RADIATION_FILE have been extended by an option to interpolate the reduced values to the regular grid. The method of the nearest neighbour is applied to each latitude.

XII.5 Example

The following tables show the land-sea mask for a regular grid (Table 14, left column) and the corresponding reduced grid (Table 15, left column). 'L' marks land- and S sea points. Both grids cover the area from 100^{0}W to 29^{0}E and from 72^{0}S to 81^{0}N. The latitude increment is 3^0 for both grids. The longitude increment for the regular grid is 3^0, and therefore in the reduced grid is 3^0 close to the equator. For each latitude the number of longitude grid points (NLON = $N_x(k)$) and the longitude increment (DELTA_LON = $d_x(k)$) are given in the right columns of the Tables. In this example the reduced grid includes 1088 sea points, compared to 1516 in the regular grid. The propagation time step in the main program Chief can be increased by a factor of ~ 4. The computational load (neglecting the source functions) is reduced by a factor of ~ 6.

Table 14: Land-sea mask for a regular grid

```
 12345678901234567890123456789012345678901234
2SSSSLSLLLLLLLLLLLLLLLLLLLLLLLLSSSSSSSSSSSSSSS
1SLLLLLLLLLLLLLLLLLLLLLLLLLLLLSSSSSSSSSLLLLSSS
0LLLLLLLSSSSSSSLLLLLLLLLLLLLLSSSSSSSSSSSSSSSSS
9LLSSSLLLLSSSSSSSLLLLLLLLLLLSSSSSSSSSSSSSSSSSS
8LLLLSSSSLLLSSSSLLLLLLLLLLLLSSSSSSSSSSSSSSLLLL
7LLLLLLSSLLLLLSSLLLLLLLSSSSSSSSSSSSSSSSSLLLLLLL
6LLLSSLSSSLLLSSSSSLLLSSSSSSSSSSSSSSSSSSLLLLSLLL
5LLSSSSSLLLSSSSSSSSLSSSSSSSSSSSSSSSSSSLLLLSSLLL
4LLLSSSSSLLLLLSSSSSSSSSSSSSSSSSSSSLLSSSSLLSSLLL
3LLLLLLSLLLLLLLSSSSSSSSSSSSSSSSSLLLLSSSSSSSLLLL
2LLLLLLLLLLLLLLSSSSSSSSSSSSSSSSSSSSLLLLLLLLLLLL
1LLLLLLLLLLLLLSSLLSSSSSSSSSSSSSSSSSSLLLLLLLLLLLL
0LLLLLLLLLLLLLSSSSSSSSSSSSSSSSSSSSSSLLLLLLLLLLS
9LLLLLLLLLLSSSSSSSSSSSSSSSSSSSSSSSLLLLSSLLSLLLSS
8LLLLLLLLLSSSSSSSSSSSSSSSSSSSSSSSSLLLLSSLSSSLSLL
7LLLLLLLLSSSSSSSSSSSSSSSSSSSSSSSSSSLSSLLLSSSSSSL
6LLLLLLLSSSSSSSSSSSSSSSSSSSSSSSSSSLLLLLLSSSSSSS
5LLLLSLSSSSSSSSSSSSSSSSSSSSSSSSSSSLLLLLLLLLSLLLL
4LSSSSSLSSSSSSSSSSSSSSSSSSSSSSSSSSLLLLLLLLLLLLLL
3LSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSLLLLLLLLLLLLLLL
2LSSSSSSLSSSSSSSSSSSSSSSSSSSSSSSLLLLLLLLLLLLLLLL
1LSLLSSSSSLSSSSSSSSSSSSSSSSSSSSSLLLLLLLLLLLLLLLL
0SSLLLLSSSSSSSSSSSSSSSSSSSSSSSSSLLLLLLLLLLLLLLLL
9SSSSLLSSSSSSSSSSSSSSSSSSSSSSSSSLLLLLLLLLLLLLLLL
8SSSSSLSSLSLLLSSSSSSSSSSSSSSSSSSLLLLLLLLLLLLLLL
7SSSSSSSSLLLLLLSSSSSSSSSSSSSSSSSLLLLSLLLLLLLLLL
6SSSSSSSSLLLLLLLLSSSSSSSSSSSSSSSSSSSSSSSLLLLLLLL
5SSSSSSSLLLLLLLLLLSSSSSSSSSSSSSSSSSSSSSSLLLLLLLL
4SSSSSSLLLLLLLLLLLLLSSSSSSSSSSSSSSSSSSSSLLLLLLL
3SSSSSSLLLLLLLLLLLLLLLLSSSSSSSSSSSSSSSSSSLLLLLLL
2SSSSSSSLLLLLLLLLLLLLLLSSSSSSSSSSSSSSSSSSSLLLLLL
1SSSSSSSLLLLLLLLLLLLLLSSSSSSSSSSSSSSSSSSSSLLLLLL
0SSSSSSSSLLLLLLLLLLLLSSSSSSSSSSSSSSSSSSSSSSLLLLLL
9SSSSSSSSSLLLLLLLLLLLSSSSSSSSSSSSSSSSSSSSSLLLLLLL
8SSSSSSSSSSLLLLLLLLLLSSSSSSSSSSSSSSSSSSSSSSLLLLLL
7SSSSSSSSSSLLLLLLLLSSSSSSSSSSSSSSSSSSSSSSSSSLLLLLL
6SSSSSSSSSSLLLLLLSSSSSSSSSSSSSSSSSSSSSSSSSSSLLLLLL
5SSSSSSSSSSLLLLLLSSSSSSSSSSSSSSSSSSSSSSSSSSSLLLLL
4SSSSSSSSSSLLLLLLSSSSSSSSSSSSSSSSSSSSSSSSSSSSLLLLS
3SSSSSSSSSLLLLLSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
2SSSSSSSSSLLLLLSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
1SSSSSSSSSLLLSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
0SSSSSSSSLLLLSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
9SSSSSSSSLLLSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
8SSSSSSSSSLLSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
7SSSSSSSSSSLSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
6SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
5SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
4SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
3SSSSSSSSSSSSLSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
2SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
1SSSSSSSSSLLLLLSSSSSSSSSSSSSSSSSSSSLLLLLLLLLLLL
 12345678901234567890123456789012345678901234
```

NO.	LATITUDE	NLON	DELTA_LON
52	+081:00:00.00	50	+003:00:00.00
51	+078:00:00.00	50	+003:00:00.00
50	+075:00:00.00	50	+003:00:00.00
49	+072:00:00.00	50	+003:00:00.00
48	+069:00:00.00	50	+003:00:00.00
47	+066:00:00.00	50	+003:00:00.00
46	+063:00:00.00	50	+003:00:00.00
45	+060:00:00.00	50	+003:00:00.00
44	+057:00:00.00	50	+003:00:00.00
43	+054:00:00.00	50	+003:00:00.00
42	+051:00:00.00	50	+003:00:00.00
41	+048:00:00.00	50	+003:00:00.00
40	+045:00:00.00	50	+003:00:00.00
39	+042:00:00.00	50	+003:00:00.00
38	+039:00:00.00	50	+003:00:00.00
37	+036:00:00.00	50	+003:00:00.00
36	+033:00:00.00	50	+003:00:00.00
35	+030:00:00.00	50	+003:00:00.00
34	+027:00:00.00	50	+003:00:00.00
33	+024:00:00.00	50	+003:00:00.00
32	+021:00:00.00	50	+003:00:00.00
31	+018:00:00.00	50	+003:00:00.00
30	+015:00:00.00	50	+003:00:00.00
29	+012:00:00.00	50	+003:00:00.00
28	+009:00:00.00	50	+003:00:00.00
27	+006:00:00.00	50	+003:00:00.00
26	+003:00:00.00	50	+003:00:00.00
25	+000:00:00.00	50	+003:00:00.00
24	-003:00:00.00	50	+003:00:00.00
23	-006:00:00.00	50	+003:00:00.00
22	-009:00:00.00	50	+003:00:00.00
21	-012:00:00.00	50	+003:00:00.00
20	-015:00:00.00	50	+003:00:00.00
19	-018:00:00.00	50	+003:00:00.00
18	-021:00:00.00	50	+003:00:00.00
17	-024:00:00.00	50	+003:00:00.00
16	-027:00:00.00	50	+003:00:00.00
15	-030:00:00.00	50	+003:00:00.00
14	-033:00:00.00	50	+003:00:00.00
13	-036:00:00.00	50	+003:00:00.00
12	-039:00:00.00	50	+003:00:00.00
11	-042:00:00.00	50	+003:00:00.00
10	-045:00:00.00	50	+003:00:00.00
9	-048:00:00.00	50	+003:00:00.00
8	-051:00:00.00	50	+003:00:00.00
7	-054:00:00.00	50	+003:00:00.00
6	-057:00:00.00	50	+003:00:00.00
5	-060:00:00.00	50	+003:00:00.00
4	-063:00:00.00	50	+003:00:00.00
3	-066:00:00.00	50	+003:00:00.00
2	-069:00:00.00	50	+003:00:00.00
1	-072:00:00.00	50	+003:00:00.00

|

Table 15: Land-sea mask for the reduced grid of the same area as in Table 14

```
 1234567890123456789012345678901234567890123456789
2SLLLLSSS
1SLLLLLSSLS
0LLSSLLLSSSSS
9LSLSSLLLLSSSSS
8LLSLSSLLLSSSSSLL
7LLLSLLLLLSSSSSSLLL
6LLLSLLSSLSSSSSSSLLLL
5LSSSLSSSSLSSSSSSSLLLLL
4LLSSLLLSSSSSSSSSSLSSLSLL
3LLLLSLLLLSSSSSSSSSLLSSSSLL
2LLLLLLLLLSSSSSSSSSSSLLLLLLLL
1LLLLLLLLSLLSSSSSSSSSSSSLLLLLLLL
0LLLLLLLLLLSSSSSSSSSSSSSSSLLLLLLLS
9LLLLLLLLSSSSSSSSSSSSSSSSSLLLSSLLLLS
8LLLLLLLSSSSSSSSSSSSSSSSSSLLSSSSSLLL
7LLLLLLLSSSSSSSSSSSSSSSSSSSSSLLSSSSSL
6LLLLLLSSSSSSSSSSSSSSSSSSSSSLLLLLSSSSSS
5LLLLLLSSSSSSSSSSSSSSSSSSSSSLLLLLLLLSLLL
4LSSSSSSSSSSSSSSSSSSSSSSSSSSLLLLLLLLLLLL
3LSSSSSSSSSSSSSSSSSSSSSSSSSSLLLLLLLLLLLL
2LSSSSSSSSSSSSSSSSSSSSSSSSSSSLLLLLLLLLLLLLL
1LSLLSSSSSLSSSSSSSSSSSSSSSSSLLLLLLLLLLLLLL
0SSLLLLSSSSSSSSSSSSSSSSSSSSSSLLLLLLLLLLLLLLL
9SSSSLLSSSSSSSSSSSSSSSSSSSSSSLLLLLLLLLLLLLLL
8SSSSSLSSLSLLLSSSSSSSSSSSSSSSSLLLLLLLLLLLLLL
7SSSSSSSSLLLLLLSSSSSSSSSSSSSSSLLLLSLLLLLLLLL
6SSSSSSSSLLLLLLLSSSSSSSSSSSSSSSSSSSSSSLLLLLLL
5SSSSSSSLLLLLLLLLSSSSSSSSSSSSSSSSSSSSSLLLLLLL
4SSSSSSLLLLLLLLLLLLSSSSSSSSSSSSSSSSSSSSLLLLLLL
3SSSSSSLLLLLLLLLLLLLLLSSSSSSSSSSSSSSSSSSLLLLLLL
2SSSSSSSLLLLLLLLLLLLLLSSSSSSSSSSSSSSSSSSLLLLLL
1SSSSSSSLLLLLLLLLLLLLSSSSSSSSSSSSSSSSSSSLLLLLL
0SSSSSSSSLLLLLLLLLLLLSSSSSSSSSSSSSSSSSSSSLLLLLL
9SSSSSSSSSLLLLLLLLLLSSSSSSSSSSSSSSSSSSLLLLLL
8SSSSSSSSSSLLLLLLLLLSSSSSSSSSSSSSSSSSSSLLLLLL
7SSSSSSSSSLLLLLLLLSSSSSSSSSSSSSSSSSSSSSLLLLL
6SSSSSSSSSLLLLLLLSSSSSSSSSSSSSSSSSSSSSSSLLLLL
5SSSSSSSSSLLLLLLSSSSSSSSSSSSSSSSSSSSSSLLLL
4SSSSSSSSLLLLLLSSSSSSSSSSSSSSSSSSSSSSSLLLS
3SSSSSSSSLLLLSSSSSSSSSSSSSSSSSSSSSSSSS
2SSSSSSSLLLLSSSSSSSSSSSSSSSSSSSSSSSS
1SSSSSSSLLLSSSSSSSSSSSSSSSSSSSSSSSSS
0SSSSSSLLLSSSSSSSSSSSSSSSSSSSSSSSS
9SSSSSSLLSSSSSSSSSSSSSSSSSSSSSSS
8SSSSSSLSSSSSSSSSSSSSSSSSSSSS
7SSSSSSLSSSSSSSSSSSSSSSSSS
6SSSSSSSSSSSSSSSSSSSSSSSSS
5SSSSSSSSSSSSSSSSSSSSSSS
4SSSSSSSSSSSSSSSSSSSSS
3SSSSSLSSSSSSSSSSSS
2SSSSLSSSSSSSSSSS
1SSSLLSSSSSLLLL
1234567890123456789012345678901234567890123456789
```

NO.	LATITUDE	NLON	DELTA_LON
52	+081:00:00.00	8	+021:00:00.00
51	+078:00:00.00	10	+016:20:00.00
50	+075:00:00.00	14	+011:18:27.69
49	+072:00:00.00	16	+009:48:00.00
48	+069:00:00.00	18	+008:38:49.41
47	+066:00:00.00	20	+007:44:12.63
46	+063:00:00.00	24	+006:23:28.70
45	+060:00:00.00	26	+005:52:48.00
44	+057:00:00.00	28	+005:26:40.00
43	+054:00:00.00	30	+005:04:08.28
42	+051:00:00.00	32	+004:44:30.97
41	+048:00:00.00	34	+004:27:16.36
40	+045:00:00.00	36	+004:12:00.00
39	+042:00:00.00	38	+003:58:22.70
38	+039:00:00.00	40	+003:46:09.23
37	+036:00:00.00	40	+003:46:09.23
36	+033:00:00.00	42	+003:35:07.32
35	+030:00:00.00	44	+003:25:06.98
34	+027:00:00.00	46	+003:16:00.00
33	+024:00:00.00	46	+003:16:00.00
32	+021:00:00.00	48	+003:07:39.58
31	+018:00:00.00	48	+003:07:39.58
30	+015:00:00.00	48	+003:07:39.58
29	+012:00:00.00	50	+003:00:00.00
28	+009:00:00.00	50	+003:00:00.00
27	+006:00:00.00	50	+003:00:00.00
26	+003:00:00.00	50	+003:00:00.00
25	+000:00:00.00	50	+003:00:00.00
24	-003:00:00.00	50	+003:00:00.00
23	-006:00:00.00	50	+003:00:00.00
22	-009:00:00.00	50	+003:00:00.00
21	-012:00:00.00	50	+003:00:00.00
20	-015:00:00.00	48	+003:07:39.58
19	-018:00:00.00	48	+003:07:39.58
18	-021:00:00.00	48	+003:07:39.58
17	-024:00:00.00	46	+003:16:00.00
16	-027:00:00.00	46	+003:16:00.00
15	-030:00:00.00	44	+003:25:06.98
14	-033:00:00.00	42	+003:35:07.32
13	-036:00:00.00	40	+003:46:09.23
12	-039:00:00.00	40	+003:46:09.23
11	-042:00:00.00	38	+003:58:22.70
10	-045:00:00.00	36	+004:12:00.00
9	-048:00:00.00	34	+004:27:16.36
8	-051:00:00.00	32	+004:44:30.97
7	-054:00:00.00	30	+005:04:08.28
6	-057:00:00.00	28	+005:26:40.00
5	-060:00:00.00	26	+005:52:48.00
4	-063:00:00.00	24	+006:23:28.70
3	-066:00:00.00	20	+007:44:12.63
2	-069:00:00.00	18	+008:38:49.41
1	-072:00:00.00	16	+009:48:00.00