A User Guide to Mexec

A Matlab-based Bespoke Processing Suite

for Ship-based Oceanographic Data

Version 4 (subset of ocp\_hydro\_matlab)

Yvonne L. Firing, N. Penny Holliday, Alejandra L. Sanchez-Franks

Ocean Circulation and Processes Subgroup

Marine Physics and Ocean Climate

National Oceanography Centre

Southampton

[1. Introduction 2](#_Toc124798237)

[1.1 About this guide 2](#_Toc124798238)

[1.2 What is Mexec? 3](#_Toc124798239)

[1.2.1 History files and data file version control for multiple people processing data 4](#_Toc124798240)

[1.2.2 Processing options and parameters 5](#_Toc124798241)

[1.2.3 Mexec conventions 5](#_Toc124798242)

[1.2.4 Mstar file format 6](#_Toc124798243)

[1.3 Changes and bugs 6](#_Toc124798244)

[2. Setting up a new cruise and using Mexec 7](#_Toc124798245)

[2.2 On the ship 8](#_Toc124798246)

[3. CTD data and water bottle sample data 11](#_Toc124798247)

[3.1 Sea Bird data acquisition and processing 11](#_Toc124798248)

[3.1.1 SBE Data Processing 11](#_Toc124798249)

[3.2 Mexec CTD data processing 12](#_Toc124798250)

[3.2.1 Output file types 12](#_Toc124798251)

[3.2.2 Processing steps to do immediately following a cast 12](#_Toc124798252)

[3.3 Water Bottle Sample Data 15](#_Toc124798253)

[3.3.1 Loading data 15](#_Toc124798254)

[3.3.2 Checking data 16](#_Toc124798255)

[3.4. Sensor Calibration 17](#_Toc124798256)

[3.5 Outputting data in other formats 17](#_Toc124798257)

[3.5.1 1hz files for LADCP processing 17](#_Toc124798258)

[3.5.2 LADCP processing for bottom depth 17](#_Toc124798259)

[3.5.3 WOCE exchange format CTD and bottle data 18](#_Toc124798260)

[3.5.3 Summary tables (\*\*\*to be updated) 18](#_Toc124798261)

[4. Underway Data 19](#_Toc124798262)

[4.1 TECHSAS/SCS/RVDAS 19](#_Toc124798263)

[4.1.1 Data access 19](#_Toc124798264)

[4.1.2 Preparation at the start of the cruise 19](#_Toc124798265)

[4.1.3 Automatic daily processing 20](#_Toc124798266)

[4.1.4 Navigation: additional processing 21](#_Toc124798267)

[4.1.5 Meteorology: additional processing 21](#_Toc124798268)

[4.1.6 Ocean surface variables: additional processing 22](#_Toc124798269)

[4.1.7 Bathymetry: additional processing 23](#_Toc124798270)

[4.2 VMADCP 23](#_Toc124798271)

[4.2.1 Acquiring and processing/editing data 23](#_Toc124798272)

[4.2.1.1 VMDAS plus CODAS 23](#_Toc124798273)

[4.2.1.2 UHDAS plus CODAS 24](#_Toc124798274)

[4.2.2 Output VMADCP station data for LADCP processing 25](#_Toc124798275)

[Appendices 26](#_Toc124798276)

[A. Quick guide to using Mexec functions and interacting with Mstar files 26](#_Toc124798277)

[B. Processing checklists 28](#_Toc124798278)

[D. Known bugs and future changes 30](#_Toc124798279)

[D.1. Bugs 30](#_Toc124798280)

# 1. Introduction

## 1.1 About this guide

This guide, and Mexec, are designed for those wishing to process CTD and underway data at sea. The underway portion is applicable to ships running SCS or TECHSAS systems. Basic familiarity with UNIX/Linux, shell scripting, and Matlab is assumed.

Throughout, > is used to indicate examples of steps run from the command line, including shell scripts, and >> for steps run from Matlab. Variables or parts of filenames that must be substituted with values are in italic (e.g. a reference to m\_daily\_proc(*day*) indicates that the day number should be substituted for *day*; opt\_*cruise*.m refers to opt\_dy113.m, opt\_jc238.m, etc.; ctd\_*cruise*\_*nnn*\_2db.nc refers to one of a set of files, with *cruise* replaced by the cruise designation and *nnn* replaced by the 3-digit form of the station number).

The following sections give instructions for setting up Mexec processing for a cruise (Section 2); processing CTD data (Section 3); and processing underway data (Section 4). A brief description of Mexec is given below, with a few more details in Appendix A, but in general this is meant to be a guide to a standard set of steps with limited variations. Example processing checklists are included in Appendix B.

This guide was first written by Penny Holliday and has been updated for v3 by Yvonne Firing and Alejandra Sanchez Franks and for v4 by Yvonne Firing.

## 1.2 What is Mexec?

Mexec is a system for processing, quality control, and integration of hydrographic data including CTD and water sample data and standard underway streams. It consists of Matlab and shell scripts, and interfaces with output of the SeaBird software (for collection and initial processing of CTD data), and with external libraries for processing of LADCP and VMADCP data. It saves its output as NetCDF files with a particular set of metadata and conventions (referred to here as Mstar files, see 1.2.3). The Matlab scripts are part of git repository git.noc.ac.uk/OCP/ocp\_hydro\_matlab, and the shell scripts are in git.noc.ac.uk/OCP/mexec\_exec.

The Mexec libraries were developed over a number of years by scientists at the UK National Oceanography Centre, Southampton, principally Brian King and Yvonne Firing, with contributions from many others including D. Desbruyeres, G. Evans, C. Florindo Lopez, N.P. Holliday, L. Houpert, E. Kent, G. McCarthy, B. Moat, A. Sanchez-Franks, D. Smeed, Z. Szuts, and E. Woodward (?I think: efw\*\*\*), as well as from external collaborators including E.P. Abrahamsen, K. Baumeister, S. Gary, and D. Ham, They use the Seawater (C. Morgan and L. Pender), GSW (SCOR/IAPSO WG127), and gamma\_n (D. Jackett and T. McDougall) libraries in calculations. They interface with the LDEO IX LADCP processing software (M. Visbeck and A. Thurnherr), and read in underway data from SCS,, TechSAS, and RVDAS (NOC/NMF) as well as VMADCP data from CODAS (University of Hawaii Currents Group).

The code has undergone numerous revisions, more recently preserved as git commits (before 2018 preserved in each cruise’s end-of-cruise backup of the processing workstation). Major changes since 2014 include:

JR15003 (2015/16): start of mexec\_processing\_scripts\_v3, with introduction of cruise options files (see 1.2.2).

JC159 (2018): introduction of git for version control, using three repositories (now superseded and no longer maintained, see below): git.noc.ac.uk/MEXEC/mexec\_processing\_scripts, git.noc.ac.uk/MEXEC/mexec, and git.noc.ac.uk/MEXEC/mexec\_exec.

JC191 (2020): revision of netcdf file-interface code to use Matlab-native functions (replacing bespoke functions developed before the Matlab-native functions were available).

JC211 (2021): addition of interface to NMF RVDAS underway data acquisition system, as well as function for saving from Matlab workspace to Mstar files using Matlab save-like interface, and (enabled by this function) the start of simplification of processing steps and reduction in number of intermediate Mstar files

JC238 (2022): continued simplification of processing steps, merge of mexec and mexec\_processing\_scripts along with other related Matlab code into a single git repository, git.noc.ac.uk/OCP/ocp\_hydro\_matlab, and shifting of (maintained) mexec\_exec repository from git.noc.ac.uk/MEXEC/ to git.noc.ac.uk/OCP/.

SD025 (2023): modularization of gridding and editing functionality; instrument serial numbers tracked; update of RVDAS interface for SDA.

EN705 (2023): update for reading NetCDF SCS underway data files.

DY174 (2024): CF-compliant time units.

Some attempts at backwards compatibility have been made, with the goal that newer code could be used to reprocess older cruises’ data (without starting completely from scratch), for instance if new bottle samples analysed ashore become available, or calibrations are reconsidered. Where file name patterns (e.g. for merged intermediate files) have changed, the aim is for code to be compatible with both old and new versions. Backwards compatibility is also maintained for some variable names but not in every case. When cruise options files were introduced on JR15003, processing choices from previous cruises were captured, and earlier cruises’ options files were generally kept up to date with changes to code through DY113. Changes from JC211 on (e.g. to option names, how they are specified, or in which scripts they are used) have been only partially propagated back to earlier cruises’ options files as of this point, so reprocessing earlier cruises’ data may require more editing of the options files.

### 1.2.1 Metadata, history files and data file version control for multiple people processing data

Each step that goes into a Mstar file is recorded both in the header comment field and in a history file for the processing stream (so, although there are multiple ctd Mstar files for each station, there will be a single ctd history file for each station). The history files are found in mexec\_housekeeping/history/. The comments in the header of the data file are set by the code writing it and, if the code is run on a Unix/Linux system, include the user and latest commit of the code used at a given step.

To prevent conflicts over modifying Mstar files, editing one also sets a lock file, in mexec\_housekeeping/version/, which will normally be reset in closing the program. If a program is interrupted mid-run, however, the flag may have to be reset manually using mreset.m.

### 1.2.2 Processing options and parameters

A number of processing parameters and variables (e.g. which CTD is primary, the calibration functions for conductivity and other parameters, etc.) change from cruise to cruise. From Mexec v3 on, rather than editing the various scripts that may contain these parameters, they are set centrally, with modifications contained in one script (per cruise), as follows: processing scripts and functions call get\_cropt.m, which

1. calls the four setdef\_cropt\_\*.m scripts to set any defaults using switch/case on two (string) variables: scriptname (generally the name of the calling script) and oopt;
2. calls opt\_*cruise*.m to set cruise-specific options, again using switch/case on scriptname and oopt; and
3. calls check\_cropt.m to (for some cases) check and warn about invalid or unset options.

The defaults are split, purely for convenience/readability, across four scripts: set\_mexec\_defaults.m contains defaults relating to CTD casts, setdef\_cropt\_sam.m those relating to sample data, setdef\_cropt\_uway.m those relating to underway streams, and setdef\_cropt\_other.m contains defaults relating to everything else (csv output files, multi-cast gridding, etc.). These setdef\*.m scripts contain explanations for each set of options and therefore may be useful to examine, but general users should not edit them. Instead, make changes to opt\_*cruise*.m for your cruise – it is recommended that you not duplicate (unchanged) default settings in this file -- and raise an issue (see 1.3) if you think the defaults themselves should change.

A guide to finding out about cruise-specific options is in Appendix A.

To determine processing choices that went into a given data file, in addition to the comment field, the metadata includes the commit active when the data file was most recently modified; the setdef\_cropt\_\*.m and opt\_*cruise*.m files in that commit thus contain a record of corresponding processing parameters. If you add parameters (e.g. a list of salinity sample flags to modify) not explicitly in opt\_*cruise*.m but by having opt\_*cruise*.m read in another file (e.g. salflags.txt), you should place that file in your data directory (e.g. in ctd/BOTTLE\_SAL) so it will be backed up along with the original input “data” files (e.g. sal\_jc238\_crate1.csv, etc.).

### 1.2.3 Mexec conventions

Mexec uses global variables for passing arguments to functions. This means if you clear the workspace you will have to re-run m\_common or m\_global before using any other Mexec programs.

The reason for this setup is that many of the original functions that performed calculations and did file i/o (now in ocp\_hydro\_matlab/file\_tools/mexec/) were designed to be run in an interactive mode, prompting for successive inputs on the command line. To call these functions within other scripts or functions, therefore, a global variable (MEXEC\_A.MARGS\_IN) is used and queried for the inputs. Because the list of inputs depends on the previous choices, the best way to figure out how to use one of these functions is to call it in interactive mode (with empty or missing MEXEC\_A.MARGS\_IN) and follow the prompts.

Partially in v3 and increasingly in v4, these functions are being superseded, first by increased calculations in the workspace, and second by functions that take input arguments in a more Matlab-standard format. This guide thus assumes users will interact with Mexec by running the wrapper scripts (ocp\_hydro\_matlab/mexec\_processing\_scripts and subdirectories, described below) and modifying the cruise options file, ocp\_hydro\_matlab/mexec\_processing\_scripts/cruise\_options/opt\_*cruise*.m, and will not need to directly call the ocp\_hydro\_matlab/file\_tools/mexec/ functions (or edit them except for some specifying underway file information, detailed in Section 4).

### 1.2.4 Mstar file format

Mexec generates NetCDF files, with a particular format (set of attributes), referred to here as Mstar files. While these NetCDF files can be read with ncdisp.m and ncread.m (or equivalent programs in other languages), there are specific Mexec scripts for interfacing with them. There are two notable idiosyncracies that may trip up when interacting with Mstar files: a) all data must be numeric; b) before DY174, their specification of time base and units is non-standard: the time units, specified as variable attributes, are seconds or sometimes days, and the origin is given as global attribute data\_time\_origin ([yyyy mm dd HH MM SS]). From DY174, CF-compliant units (e.g. ‘seconds since 2024-01-01 00:00:00’) are used in time variable units attribute, and the data\_time\_origin global attribute can be ignored.

A quick guide to interacting with Mstar files is given in Appendix A. For python users loading older (pre-DY174) Mstar data into xarray, you may need to force the time variable \*not\* to be read not as a datetime, and then convert it.

## 1.3 Changes and bugs

We encourage you to create a pull request or send us your scripts at the end of your cruise (ideally with context about your changes or additions as an issue or by email to [yvonne.firing@noc.ac.uk](mailto:yvonne.firing@noc.ac.uk)), so that other users can benefit from your improvements and examples. To make it easier for us to track and integrate changes, please use ocp\_hydro\_matlab/ only for scripts/functions that you think should be maintained in the code going forward (e.g. a bug fix to an existing script, or a new script to read in a new type of data that may be used again); use a different directory for working code just for your cruise (e.g. quick code making plots of data specifically for your cruise).

# 2. Setting up a new cruise and using Mexec to process data

## 2.1 Before the cruise

1. Get the ocp\_hydro\_matlab git repository.

If you have git, clone the main branch:

> cd ${PROGDIR} #this is wherever you choose to keep the software

> git clone [git@github.com/NOC-OCP/ocp\_hydro\_matlab.git](mailto:git@github.com/NOC-OCP/ocp_hydro_matlab.git) #if you have a github login

OR

> git clone <https://github.com/NOC-OCP/ocp_hydro_matlab.git>

Generally the main branch is the version you should use, but in special cases you may instead want to work from a different cruise branch:

> cd ${PROGDIR}/ocp\_hydro\_matlab

> git branch *other\_branch*

> git fetch origin *other\_branch*

> git checkout origin *other\_branch*

> git pull origin *other\_branch*

Otherwise, download the desired branch (probably the master branch) from <http://github.com/NOC-OCP/ocp_hydro_matlab> and unpack in ${PROGDIR}.

1b) If you intend to use the shell scripts for syncing data from a server to your Linux/MacOS working machine, also clone or download (as above) the mexec\_exec git repository from git.noc.ac.uk/OCP/mexec\_exec.

1. Determine your cruise designation, typically 2-3 letters denoting the ship and 3-5 letters denoting the cruise number, e.g. dy113, jr18002, nbp0705. This will be used in various places and is the variable meant by (italic) *cruise* through the rest of this document.

If you are using git, for each of the repositories, create and switch to a new branch for your cruise:

> cd ${PROGDIR}/ocp\_hydro\_matlab

> git branch *cruise*

> git checkout *cruise*

If you are using mexec\_exec also:

> cd ${PROGDIR}/mexec\_exec

> git branch *cruise*

> git checkout *cruise*

1. Add the full path to ocp\_hydro\_matlab to your Matlab startup path.
2. If using mexec\_exec:
   1. Add it and its subdirectories to your shell path (in .bashrc or equivalent).
   2. Edit the cruise name and processing base directory (e.g. /local/users/pstar/dy113/mcruise/) in ${PROGDIR}/mexec\_exec/conf\_scripts/conf\_script\_mexec. You may also need to edit the underway data system and a flag for whether LADCP data are acquired. Run to configure the cruise directory structure:

> conf\_script\_mexec

This will set up a processing directory structure, including relative symbolic links so that the directory structure can be copied elsewhere (e.g. at the end of a cruise) and processing continued.

1. Configure ocp\_hydro\_matlab:
   1. Add the full path to ocp\_hydro\_matlab/ to your Matlab startup path
   2. Edit ocp\_hydro\_matlab/mexec\_processing\_scripts/m\_setup.m. Most things you (may) need to modify are near the top of the file, including *cruise* (stored as MEXEC\_G.MSCRIPT\_CRUISE\_STRING), and the paths to other software used in processing (e.g. seawater toolboxes). You can alternately specify the *cruise* and a few other variables in an input argument to m\_setup.
   3. Generate an empty cruise-specific options file, ocp\_hydro\_matlab/mexec\_processing\_scripts/cruise\_options/opt\_*cruise*.m, replacing *cruise* with your cruise reference (in the same form as you set in m\_setup.m, e.g. dy113). See below, as well as other files in that directory, for examples.

## 2.2 On the ship

6) Remotely mount data filesystems, and if necessary set up symbolic links, e.g.

> ln -s /mnt/Data/current\_cruise ~/mounts/mnt\_cruise\_data

The NOC OCP seagoing workstations have mount points for Cook and Discovery in /etc/fstab and preserved under ~/mounts/, e.g. \*\*\*

7) Edit shell scripts in mexec\_exec/: ctd\_syncscript, lad\_syncscript (for LADCP), and uhdas\_01\_linkmerge (if reading UHDAS/CODAS data).

These scripts will be used to sync data from the remote directories to the processing directory structure, and will need to be edited to reflect file naming conventions in the remote directories (e.g. ~/mounts/CTD/Data/CTD\_DY113\_001.cnv vs. ~/mounts/CTD/Data/Processed/ctd\_dy113\_001\_01.cnv, etc.).

You may also want to use mexec\_exec/cruise\_backup to sync the processing directory and scripts to an external hard drive; in this case, edit the directories set near the top of cruise\_backup, and (recommended) add it to your crontab file to run regularly.

8) Set up or link underway data processing

a) TECHSAS or SCS: edit ocp\_hydro\_matlab/file\_tools/mexec/mtechsas/mtnames.m (for TECHSAS) or ocp\_hydro\_matlab/file\_tools/mscs/msnames.m (for SCS): add or comment/uncomment lines as necessary to reflect the stream names available on your cruise. If adding a new type of stream you can decide on the Mexec abbreviation.

b) RVDAS: get address login credentials for rvdas database machine. Set up ~/.pgpass file containing:

server:port:database:username:password

Run mrjson\_get\_list.m. Consider adding things to skip under mrvdas\_ingest case in setdef\_cropt\_uway.m. look at mrtables\_from\_json.m and see if there’s anything to add to mrnames.m (new streams) which is a lookup between shortnames (output file prefixes) and the streamnames (instrument\_talkIDmsgID). If so you may also need to add it to muwaydirs.m, which has a list of shortnames (first column) and directories to put them in. Then, edit opt\_*cruise*.m to delete the line about skipunderway, edit m\_setudir.m and rerun m\_setup.m. That should run m\_setudir.m and create some subdirectories, e.g. nav/pmv, nav/sea, met/sonic, etc. It will also output a list of streamnames that it doesn’t know where to put, and of directories it’s expecting but doesn’t find any table for.

On any system: You may need to edit and force rerun m\_setudir.m once the cruise starts. If you added new Mexec stream abbreviations above, add them and the directories where you wish those streams to be processed to the list in m\_setudir.m. In any case you may need to uncomment/comment out newly relevant/irrelevant lines.

If new underway streams become available during the cruise, remove ocp\_hydro\_matlab/mexec\_processing\_scripts/underway/m\_udirs.m and regenerate it, and make the new directories, by running m\_setudir.m.

9) Edit template files in ocp\_hydro\_matlab/mexec\_processing\_scripts/varlists/ (if necessary)

Template files are used to control lists of variables within scripts. They include ctd\_renamelist.csv, sam\_varlist.csv, dcs\_varlist.csv, and cchdo\_varlist.csv and cchdo\_ctd\_varlist.csv which determine lists of variables to be loaded for CTD, bottle sample, and other files, and how (if) they will be renamed. For SCS ships, there is also the set of scs\_renamelist\_source.csv files.

The list of variable names that you require in each file will vary from cruise to cruise depending on which samples are being collected. The ctd\_, sam\_, and cchdo\_ template files contain many possible variables, so in most cases you will just need to delete lines. The scs\_ files may need to be edited but most likely not.

ocp\_hydro\_matlab/mexec\_processing\_scripts/varlists/mcvars\_list.m: make sure the two lists in this file include all the variables you want to carry through CTD processing and sample comparison, respectively. It is not necessary to comment out variables you don’t have.

## 2.3 Using Mexec

Each time Matlab is started, run m\_setup to initialize the environment for Mexec processing by adding paths and generating global variables. If you clear all variables at any point, run m\_common or m\_global to regenerate them.

# 3. CTD data and water bottle sample data

## 3.1 Sea Bird data acquisition and processing

The first step is to select the SBE output variables in the SBE data acquisition software, SeaSave. Record the setup by saving a .XMLCON file. It is essential that the output variables include scan and pressure temperature, and it is highly useful for them to include NMEA latitude and longitude if those streams are available (generally the case on modern research vessels). For some variables (e.g. turbidity), the conversion from voltage to physical units may result in loss of precision, so better results may be obtained by outputting the raw voltage stream.

Here is an example from JC086.

# name 0 = timeS: Time, Elapsed [seconds]

# name 1 = depSM: Depth [salt water, m]

# name 2 = prDM: Pressure, Digiquartz [db]

# name 3 = t090C: Temperature [ITS-90, deg C]

# name 4 = t190C: Temperature, 2 [ITS-90, deg C]

# name 5 = c0mS/cm: Conductivity [mS/cm]

# name 6 = c1mS/cm: Conductivity, 2 [mS/cm]

# name 7 = sal00: Salinity, Practical [PSU]

# name 8 = sal11: Salinity, Practical, 2 [PSU]

# name 9 = sbeox0V: Oxygen raw, SBE 43 [V]

# name 10 = sbeox0Mm/Kg: Oxygen, SBE 43 [umol/Kg]

# name 11 = sbeox0ML/L: Oxygen, SBE 43 [ml/l]

# name 12 = xmiss: Beam Transmission, Chelsea/Seatech/WET Labs CStar [%]

# name 13 = flC: Fluorescence, Chelsea Aqua 3 Chl Con [ug/l]

# name 14 = turbWETbb0: Turbidity, WET Labs ECO BB [m^-1/sr]

# name 15 = altM: Altimeter [m]

# name 16 = scan: Scan Count

# name 17 = ptempC: Pressure Temperature [deg C]

# name 18 = pumps: Pump Status

# name 19 = latitude: Latitude [deg]

# name 20 = longitude: Longitude [deg]

# name 21 = flag: 0.000e+00

For combining CTD data with Niskin bottle sample data, the CTD data file names should incorporate a unique integer cast number, referred to here as station number – not to be confused with a number or alphanumeric designating a planned station/site. (For instance, imagine a cruise starts with a test CTD followed by a full CTD both at site A-55, and then a CTD at site A-53: the cast or station numbers for Mexec processing could be (1, 2, 3) or (990, 1, 2) or even (10, 9, 15) – but not 55, 55, 54, nor 1.1, 1.2, 2.)

## 3.1.1 SBE Data Processing

On the CTD logging computer, the SBE Data Processing software should be used for initial processing when the cast is finished. First run:

1. Data Conversion to convert the raw frequency and voltage data to engineering units as appropriate by applying the manufacturer's calibrations stored in the CON file and saving both downcast and upcast to an ASCII format (.cnv) file.

* This step may include oxygen hysteresis correction using SBE default parameters, but we recommend not applying the correction here but instead applying it in mexec processing (this makes it easier to change the parameters if necessary). If you decide to correct for oxygen hysteresis at this stage you will need to change the dooxyhyst flag in opt\_*cruise*.m).

The output file names should contain the three-digit sequential station/cast number, and should not have spaces; otherwise their form is not important. Add the filename (and location) to the ctd\_proc, cnvfilename case in opt\_*cruise*.m.

The next two steps can be run either in SBE Data Processing or in mexec:

1. Align CTD to align the oxygen sensor in time relative to pressure. Recommended: set the output name to \_align so that, for input CTD\_CRUISE\_nnn.cnv, this step will produce CTD\_CRUISE\_nnn\_align.cnv.
2. Cell Thermal Mass to correct the pressure and conductivity. Recommended: set the output name to \_ctm so that, for input CTD\_\_CRUISE\_nnn\_align.cnv, this step will produce CTD\_­CRUISE\_nnn\_align\_ctm.cnv.

The first and last .cnv files (that is, original and \_align\_ctm), as well as the .bl and .ros files, should be copied to /local/users/pstar/cruise/data/ctd/ASCII\_FILES, while .hex, .hdr, and .XMLCON files should be copied to RAW\_CTD\_FILES. On unix/linux systems you can use (mexec\_exec) ctd\_syncscript to do this after editing cruise name and location of original files.

## 3.2 Mexec CTD data processing

### 3.2.1 Output file types

ctd\_*cruise*\_*nnn*\_\*.nc contain CTD time series or profiles, with different stages of processing, editing, and averaging (see below).

dcs\_*cruise*\_*nnn*.nc contains information about scans (start, bottom, end of cast) and positions

fir\_*cruise*\_*nnn*.nc contains information about bottle firing times and corresponding CTD and winch data

sam\_*cruise*\_all.nc is a combined (all-station) file containing the data from the fir\_*cruise*\_*nnn*.nc files along with analysed bottle sample data.

win\_*cruise*\_nnn.nc contains winch information (wireout, tension).

### 3.2.2 Processing steps to do immediately following a cast

The basic steps for CTD processing following a cast (see Appendix B) are:

> ctd\_syncscript

Copies .hex, .cnv, \_align\_ctm.cnv, and .bl files from acquisition computer to processing workstation.

>> stn = *n*; ctd\_all\_part1 % *n* is the integer station number

ctd\_all\_part1.m calls the following:

mctd\_01.m loads and renames variables (as set in varlists/ctd\_renamelist.csv and cruise options files) and saves in ctd\_*cruise*\_*nnn*\_raw.nc.

mctd\_02.m does conversions, edits, and corrections as set in cruise options files, calling ctd\_apply\_autoedits.m, ctd\_apply\_oxyhyst.m, select\_calibrations.m, and apply\_calibrations.m to produce ctd\_*cruise*\_*nnn*\_raw\_cleaned.nc and ctd\_*cruise*\_*nnn*\_24hz.nc files. The only default action is to apply (the manufacturer default) correction for oxygen hysteresis, but examples are available for code to remove out of range values, certain scan ranges, spikes, and/or times when the pumps were off; it is generally not recommended to apply these before first examining data (as problems may be masked). Once sample data are available, the mctd\_02 case of the cruise options file is also where calibration functions can be specified (with a setting to apply to all stations or only a subset).

mctd\_03.m selects primary sensors (as set in cruise options files), computes derived variables (e.g. salinity) and averages to 1 Hz, using grid\_profile.m, producing ctd\_*cruise*\_*nnn*\_psal.nc.

mdcs\_01.m guesses start and bottom of cast and saves in dcs\_*cruise*\_*nnn*.nc.

>> stn = *n*; mdcs\_03g

mdcs\_03g.m brings up a GUI for selection or confirmation of cast start, bottom, and end based on P, T, C, and pumps flag; any modifications are added to dcs\_*cruise*\_*nnn*.nc.

For the start of the downcast, select the lowest pressure after the CTD has soaked and been brought to the surface before descending (unless it was brought too close to the surface, causing erroneous conductivity values, in which case, select the start of the good data). For the end of the upcast, select the last scan for which there was good in-water oxygen, temperature, conductivity and salinity data (note that oxygen data becomes out-of-water before the other variables because of the different sensor response times).

Edit opt\_cruise.m if necessary: if upon sampling some Niskins were leaky or questionable, edit their flags; if the cast was not full-depth add a nominal water depth (see Appendix A for how to find out where).

>> stn = *n*; ctd\_all\_part2

ctd\_all\_part2.m calls the following:

mctd\_04.m separates down and up casts, optionally applies m\_loopedit.m to the downcast data, and averages to 2 dbar, producing ctd\_*cruise*\_*nnn*\_2db.nc and ctd\_*cruise*\_*nnn*\_2up.nc.

mfir\_01.m gets times and scans of Niskin bottle firing from .bl file, along with Niskin bottle numbers (default: 1:24) and flags (default: 2 for all fired bottles, 9 otherwise) from cruise options files, and puts in fir\_*cruise*\_*nnn*.nc.

mfir\_03.m gets CTD data from these scans from the 1-Hz \_psal.nc file and adds to fir\_*cruise*\_*nnn*.nc.

mwin\_01.m gets winch information from the underway stream and saves in win\_*cruise*\_*nnn*.nc.

mwin\_to\_fir.m adds the winch information to fir\_*cruise*\_*nnn*.nc.

mfir\_to\_sam.m puts the data from fir\_*cruise*\_*nnn*.nc into appended sam\_*cruise*\_all.nc.

station\_summary.m adds position, start and end time, depth obtained by calling best\_station\_depths.m, and information from sam\_*cruise*\_all.nc for this station to station\_summary\_*cruise*.nc and table station\_summary\_*cruise*.txt. Rather than storing station depths in a text file, they are calculated by best\_station\_depths.m. Immediately following a cast, it will try to calculate them from CTD+altimeter; later you can specify to include information from the LADCP (if available), but in any case, if you have casts that are not full-depth, you should use the underway bathymetry data to fill in a list of bottom depths under the best\_station\_depths case in opt\_*cruise*.m.

mdep\_01.m adds the depths to the various ctd\_*cruise*\_*nnn*\_\*.nc files.

get\_sensor\_groups.m adds serial numbers to the sample file.

>> stn = *n*; mctd\_checkplots

mctd\_checkplots.m produces a set of plots to check for sensor drift or other problems. It will make plots comparing the two sensors, plots comparing up- and downcast data. It will also make plots comparing station *n* with other stations, first querying for either some number of preceding stations (including 0) or a list of specific station numbers to use. If you notice loops/static instabilities here, you can activate the cruise option to apply loop editing in mctd\_04.m.

>> stn = *n*; mctd\_rawshow

mctd\_rawshow.m allows inspection of 24 Hz data. If this reveals editing needed (e.g. spikes that are large enough to affect averaged data), there are normally two options:

1. specify automatic edits (based on scan ranges, data ranges, behaviour if pumps go off, and despiking) in opt\_*cruise*.m under mctd\_02.m case; and/or
2. >> stn = *n*; mctd\_rawedit

mctd\_rawedit.m brings up a GUI for selecting and deleting spikes in the data (starting from ctd\_*cruise*\_*nnn*\_raw\_cleaned.nc if available, ctd\_*cruise*\_*nnn*\_raw.nc otherwise), and saves the selections to ctd\_*cruise*\_*nnn* \_raw\_cleaned.nc, as well as recording them in (text file) mplxyed\_*yyyymmdd*\_*HHMMSS*\_ctd\_*cruise*\_*nnn*

If there are significant instrument or cable problems, the inspection may show spikes in pressure or temperature that are large enough to affect the other data streams, requiring reprocessing from before the cellTM stage\*\*\* details in \*\*\*.

If automatic edits were added or mctd\_rawedit was used,

>> stn = *n*; ctd\_all\_postedit

ctd\_all\_postedit.m reruns mctd\_02, mctd\_03, mctd\_04, mfir\_03, and mfir\_to\_sam to apply edits and propagate changes to the ctd\_*cruise*\_*nnn*\_raw\_cleaned.nc file through to other files.

If necessary, iterate the rawedit and postedit steps, and/or mctd\_checkplots.m, ctd\_all\_part2.m to loopedit.

Note: If you need to restart processing from the mctd\_01 stage after doing manual edits in mctd\_rawedit, to reapply them run ctd\_apply\_autoedits.m.

## 3.3 Water Bottle Sample Data

### 3.3.1 Loading data

The result of the sample data processing is to create a master sample data file, sam\_*cruise*\_all.nc, populated with CTD firing data, sensor data, and subsequently the water sample data as they become available. The CTD winch, firing and sensor data are pasted into this file during running of ctd\_all\_part2.m, as described above. This section describes how the water sample data are included in the process.

To link analysed water sample data with the corresponding CTD data (and with each other) we use CTD cast number (referred to here as station number, or field statnum in the Mstar files) and Niskin position (i.e. the carousel position, or field position in the Mstar files). The two parameters can be combined to make a unique “sample number” variable, e.g.:

>> sampnum = d.statnum\*100 + d.position;

Scripts including msal\_01.m, moxy\_01.m, mnut\_01.m, mco2\_01.m, mcfc\_01.m, miso\_01.m perform the following steps, depending on parameters and code specified in opt\_*cruise*.m:

1. Read in sample data from excel (single or multiple sheet) or ascii csv files (with or without header lines preceding one or more column header rows and zero or more units rows) or from netcdf or Matlab files. Currently, for excel/csv files, every data row is required to have a value for every column (e.g. if station number is a column, even if the file contains data from only a single station, the number must be filled in on each line). For samples collected but not yet analysed, the “data” is a record of where (cast and Niskin) they were collected, and goes into a *parameter*\_flag field for reference.
2. Map variable (or column) names to standardized names used in the Mexec processing and Mstar files, and check units (if supplied) or add them (if not).
3. Calculate additional quantities and apply corrections, e.g. calculate oxygen concentration from titre, standard, blank, and draw temperature values; apply salinity standards offsets and average salinity readings; modify flags based on opt\_*cruise*.m
4. Save data to *type*\_*cruise*\_*nnn*.nc file(s), where type is e.g. sbe35, sal, oxy, etc., and to sam\_*cruise*\_all.nc. The individual data type files contain more information on each than the concatenated sam file. For instance, for oxygen the replicate values and their individual flags will be recorded (as botoxya, botoxyb, etc. and botoxya\_flag, botoxyb\_flag, etc.) in oxy\_*cruise*\_*nnn*.nc, but only the average (of good values) and correspondingly updated flags will be copied to sam\_*cruise*\_all.nc. Before saving to sam\_*cruise*\_all.nc, flags will additionally be updated for consistency with Niskin bottle flags (i.e., if a Niskin has been flagged as 4 or 3 for having closed at the wrong depth or leaked, respectively, all samples from that bottle will be flagged 4 “bad”). For salinity, only CTD samples are copied to sam\_*cruise*\_all.nc, while TSG samples are written to tsg\_*cruise*\_all.nc.

The key information to determine early in the process is:

1. The format and column headers/variable names of the input sample files
2. whether information from file headers is required (e.g. salinometer bath temperature and standard seawater K15 value) and can be read in or should be coded into opt\_*cruise*.m.
3. whether you will want to recalculate variables or whether concentrations and salinities will be provided as-is by analysts. For salinity, it is recommended that you read in conductivity ratio values for both samples and standards, and inspect them (using msal\_01.m) before coding and applying standards offsets and then calculating conductivity and salinity.

Thinking through these questions early on can help to not only make the files easier to parse (e.g. by making sure they do include columns for CTD station and Niskin, or that standards and samples are clearly denoted) but also to make sure that all required information is collected (e.g. lab temperature) and that everyone is using the same convention for any flags supplied (e.g. WOCE flags: readthedocs.io/exchange\_format).

### 3.3.2 Checking data

As sam\_*cruise*\_all.nc gathers all the sample and corresponding CTD data, you can inspect and compare them however you like. Several functions are available to facilitate checking various data quantities against each other, neighbouring profiles, and/or the CTD. It is generally helpful to run them iteratively as the cruise progresses.

checkbottles\_01 plots a single sample parameter for multiple stations, as well as its anomaly (either from the mean, or, if a gridded section file, ctd/grid\_cruise\_\*.nc, has been produced, from the gridded profile), as a function of pressure and as a function of temperature; a graphical user interface enables selection and printing of outliers to be flagged in opt\_*cruise*.m or investigated further.

checkbottles\_02 plots multiple parameters for a single station, overlaying sample values on CTD or gridded (see above) profiles and labelling Niskin numbers. This script is particularly useful for detecting leaking or closed-at-the-wrong-depth Niskins, which will exhibit bad values for multiple parameters (including, potentially, obvious outliers in oxygen draw temperature).

mctd\_evaluate\_sensors compares data from a given CTD sensor (e.g. cond1, oxygen2, temp2, or can specify by serial number) to calibration (bottle sample, or for temperature, SBE35) data and to data from the other CTD, as functions of station number (a proxy for time), pressure, temperature, and the parameter itself, to enable the user to get a sense of how the sensors are behaving, look for drifts, pressure dependencies, scale factors, etc., and propose a calibration function to correct the CTD data. Calibration functions are coded in opt\_*cruise*.m, and a switch allows mctd\_evaluate\_sensors to apply them to test out. The script also has the option to plot large-residual sample and CTD values against individual station profiles, to check whether they represent bad sample analyses, or simply larger variance in properties at a given bottle stop (e.g. in a high gradient region).

Sample flags set in opt\_cruise.m under msal\_01 etc. cases, and/or Niskin flags set under the mfir\_01 case, can be updated to reflect problems identified here. Flags are then applied to the sam\_*cruise*\_all.nc file by rerunning the script under whose case they were set (e.g. msal\_01, mfir\_01).

Flag meanings for Niskin bottles (mfir\_01): initial flags should be 2 (bottle good, sampled), 3 (leaking [and did not sample]), 4 (did not trip correctly [wire not released or bottle was seen to snap closed on landing]), 7 (unknown problem [maybe leaking but not so obviously we didn’t sample]), 9 (did not sample, no further information). Then after examining data update initial flags of 7 to 2 (good), 3 (several sample values suspicious so probably leaking), or 4 (clearly sample is from a different depth, i.e. did not trip correctly), and add additional 3 or 4 flags as indicated by the data.

## 3.4. Sensor Calibration

Calibration functions are entered in opt\_*cruise*.m under the mctd\_02, ctd\_cals case. They are entered as strings which describe setting dcal.*sensor* (e.g. dcal.cond1, dcal.oxygen2, or can be specified by serial number) as a function of d0.*sensor*, as well as (optionally) d0.statnum, d0.press, d0.temp[1/2] etc. Each string is the value of a field whose name is the cruise name (to prevent applying copy-pasted calibration functions from a previous cruise). Additionally, flags under this case must be set to 1 in order for the calibrations to be applied when mctd\_02 is run. More details are given in set\_mexec\_defaults.m (under mctd\_02, ctd\_cals).

It is beneficial to calibrate temperature (if comparison data are available) before choosing a calibration for conductivity. Mctd\_evaluate\_sensors (see previous section) can help with testing the calibrations entered into opt\_*cruise*.m.

## 3.5 Outputting data in other formats

### 3.5.1 1hz files for LADCP processing

### 3.5.2 LADCP processing for bottom depth

To run basic processing of LADCP data from cast nnn (after mout\_1hzasc has been run):

> lad\_linkscript\_ix # to copy data from network machine

>> cd ladcp/ix

>> cfgstr.orient = ‘DL’; process\_cast\_cfgstr(nnn, cfgstr);

This will generate plots as well as matlab files in ladcp/ix/DL\_GPS/processed/nnn/

And if you have dual instruments, you can process the uplooker on its own:

>> cfgstr.orient = ‘UL’; process\_cast\_cfgstr(nnn, cfgstr);

And both together:

>> cfgstr.orient = ‘DLUL’; process\_cast\_cfgstr(nnn, cfgstr);

If you have the CTD 1 Hz file, you can include bottom tracking as a constraint:

>> cfgstr.orient = ‘DL’; cfgstr.constraints = {‘BT’}; process\_cast\_cfgstr(nnn, cfgstr);

SADCP data can also be used (see below).

### 3.5.3 WOCE exchange format CTD and bottle data

mout\_exch\_sam.m and mout\_exch\_ctd.m, respectively, write bottle sample and corresponding CTD data from sam\_cruise\_all.nc, and 2-dbar downcast CTD profiles from ctd\_cruise\_nnn\_2db.nc, to WOCE exchange format (ascii) files. mout\_exch\_ctd.m writes one station/file at a time. File headers are customized in opt\_cruise.m; the header information should include a note on which quantities are calibrated (and how, including salinity standard batch number) and which are not.

### 3.5.3 Summary tables (\*\*\*to be updated)

station\_summary.m produces a table of CTD casts, with columns including start, bottom, and end times; depth; number of bottles fired; number of salinity samples; and numbers of other samples, customized in opt\_cruise.m

tsg\_summary prints out/makes plots of some info from merged files (not sure this one works, some of the input files may not be current)

sam\_listing is a function that just prints the CTD data from bottle firing times for a particular station

mout\_sam\_csv

mctd\_makelists

# 4. Underway Data

## 4.1 TECHSAS/SCS/RVDAS

### 4.1.1 Data access

Mexec uses 'short names' to access the TECHSAS and SCS streams or RVDAS tables through lookup tables set in mtnames.m, msnames.m, or mrnames.m, respectively. Additional lines can be added to these files, and irrelevant lines commented out, as necessary.

The following Mexec Matlab commands can be used for a quick look at RVDAS data (substitute mt or ms in place of mr for TECHSAS or SCS respectively).

help mrvdas lists the ‘mr’ commands (alternates: mtechsas or mscs)

mrlookd tells you filename, start, end

mrnames lists mexec 'shortnames', full filenames in cell array.

mrdfinfo winch provides info about that datastream (eg winch)

mrgaps gyro\_s 10s lists gaps in datastream of more than 10s

mrposinfo([yyyy mm dd hhmm]) gives you position for that time

mrlistit list segments of data

mrload load data from specified table/stream and time range

### 4.1.2 Preparation at the start of the cruise

RVDAS

1. Install postgresql (on Mac: find out where command-line psql is on your path (possibly under /Library/ ) and make /usr/local/bin/psql a symbolic link to it (alternately, edit the line in m\_setup.m that sets MEXEC\_G.RVDAS.psql\_path).
2. If necessary, edit the lines in m\_setup.m setting MEXEC\_G.mexec\_data\_root, MEXEC\_G.mexec\_source\_root, and MEXEC\_G.other\_programs\_root to where: you want the processed data; ocp\_hydro\_matlab; and the directory containing seawater and gsw toolboxes. Make sure that in the data processing directory (e.g. ~/cruise/data/) you have a subdirectory rvdas containing directories json\_files/ and rvdas\_csv\_tmp/ (the rest will be generated automatically).
3. Determine the address (name or ip) and login credentials of the RVDAS server, as well as the location of its .json files and the database name for your cruise. Add the address, database, and username to m\_setup.m, and to file ~/.pgpass, a text file containing one or more lines specifying machine:port:database:username:password. Add the directory containing the .json files (on the remote/server machine) to m\_setup.m under MEXEC\_G.RVDAS.jsondir. (If for some reason the files are not on the same machine as the database, see alternate B below). Also, in opt\_da001.m, change skipunderway to 0.
4. Run mrjson\_get\_list.m, editing the list of files to exclude instruments you’re not interested in. [Alternate B: If necessary, copy over the .json files and make a list file (containing full paths) manually and just do the last couple of lines of mrjson\_get\_list.m where it calls mrjson\_load\_all.m, with outfile being mrtables\_from\_json.m (with full path).] At this point you should be able to run mrlistit, mrlast, and mrload. You can also run mrdefine, which will output the list of tables and variables in the matlab workspace.
5. Inspect mrtables from\_json.m and if necessary add instruments/tables/messages/patterns to exclude to setdef\_cropt\_uway.m or (probably better) opt\_*cruise*.m (under case mrvdas\_ingest, oopt *your ship* [e.g. sda]), and/or add more code for parsing variables/units to mrrename\_tables.m, and/or add new streams and give them shortnames in mrnames.m (and if you have added new shortnames, also add directories for them in muwaydirs.m).
6. If you’ve added to setdef\_cropt\_uway.m/opt\_*cruise*.m, rerun mrjson\_load\_all.m to regenerate mrtables\_from\_json.m (with more variables commented out etc.).
7. Run mrdefine, inspecting lists of original tables and renamed tables, and iterate if necessary.
8. Find and remove m\_udir.m and then run m\_setup.m to regenerate it.
9. Set MEXEC\_G.default\_navstream and MEXEC\_G.default\_hedstream in m\_setup.m to point to the shortnames of the best position and heading streams. Possibly edit munderway\_varname to add new names for lon, wind speed, etc. (alternately, you can make sure they match what was used last time by editing mrrename\_tables.m, but if you want to keep different names, perhaps for comparison of different sources of wind etc., add the alternate possibilities in munderway\_varname; various scripts call this and select the first match they find, so for instance any of lon, long, and longitude will be treated the same, etc.).
10. (Optional): compare table described in the json files with tables actually present in rvdas:

js = mrshow\_json\_all(‘~/cruise/data/rvdas/original\_json\_files/list\_json.txt’);

jslist = fieldnames(js);

whos jslist

t = mrgettables;

tlist = fieldnames(t)

1. If any data streams are reported in voltage and require factory calibration coefficients to convert to physical units, enter these in opt\_*cruise*.m under mday\_01\_clean case.

### 4.1.3 Automatic daily processing

Standard underway data (including navigation, surface air and water, and bathymetry) can be processed on a day-by-day basis by running

>> days = [ddd]; uway\_daily\_proc

where days is a vector of the yeardays you want to process, not exceeding yesterday (the last complete day). If days is not found it defaults to yesterday.

uway\_daily\_proc.m:

* goes through the list of underway data streams found in mrnames (or mtnames/msnames) finds which ones are present in the rvdas table list or the scs or techsas link directory, and calls mday\_01.m to load them, producing a series of daily files from each data stream, located in their individual directories (e.g. bathy/singleb/singleb\_jc238\_d197\_raw.nc). It is also possible to exclude some streams/set a limited list of streams for uway\_daily\_proc.m, but by default it processes everything found in both mrnames (etc.) and m\_udir.m. If no data are available on a day from a given stream, that stream is skipped.
* performs additional processing and cleaning steps on some streams by calling mday\_01\_clean\_av.m, which has cases for different streams and calls various scripts to do automatic processing steps including renaming variables to standard names (e.g. head\_gyr, depth), correcting (especially in TECHSAS) or standardising units strings, applying factory calibration coefficients (i.e. to convert from voltage to physical units) where necessary, searching for and flagging backwards time steps or duplicate times in nav streams, NaNing out-of-range values, correcting echosounder depth for speed of sound variations based on the Carter tables, and averaging bathymetry to 30-s; output files are stream\_*cruise*\_d*ddd*\_edt.nc.
* For bathymetry, when both the singlebeam (EA600/EA640) and multibeam (EM120/EM122) centre beam streams are present, mbathy\_av\_merge.m is called to paste in the depths from the other instrument for subsequent comparison.
* After all days have been loaded and edited, calls m\_uway\_append.m to add the days’ files to the appended file for each data stream (e.g. sim\_cruise\_01.nc). The list of daily files appended into the master files is given in the header information of that file.
* Finally, calls mnav\_best.m, mwind\_true.m, mtsg\_medav\_clean\_cal.m, and (for scs) upate\_allmat.m, which calculate new variables from the appended files.

The following sections contain further details of the individual data streams, manual quality control/editing steps, and the final steps operating on the appended files.

### 4.1.4 Navigation: additional processing

The navigation streams in whose position and heading we have most confidence are set in m\_setup.m. Script mnav\_best.m loads these appended files and averages and merges to a common, 30-s time base, producing a file bst\_*cruise*\_01.nc containing position, heading (using proper vector averaging), course and speed over ground, and distance run. This is the ‘definitive’ cruise navigation file.

Navigation streams, including different sources of GPS position, do not always agree to within their stated accuracies (see e.g. DY146 cruise report), so it is worth comparing the different streams carefully especially if they are being used for e.g. ADCP processing or wind calculations.

### 4.1.5 Meteorology: additional processing

Ship speed, position and heading from the bst navigation file are merged onto the wind data in the surfmet file. The absolute wind speed is calculated and vector averaged in one multi-step script mwind\_true.m. The output files from this processing are  
surfmet\_*cruise*\_true.nc  
surfmet\_*cruise*\_trueav.nc  
The latter file is reduced to 1-minute averages, with correct vector averaging when required. In order to avoid ambiguity, variable units are explicit in whether wind directions are ‘towards’ or ‘from’ the direction in question. The result is a bit cumbersome, but should be unambiguous if the units are read carefully.

Note: TECHSAS stores wind speed in m/s, but says the variable unit is knots. This is corrected in mday\_01\_clean\_av.

Wind over the stern: The standard test of whether the relative wind processing has been done correctly would be to observe no change in the calculated absolute wind when the ship changes direction or speed. This can be misleading, since the anemometer sited on the foremast under-reads speed by a significant margin when the wind is over the stern. Therefore if either the ‘before’ or ‘after’ wind direction is over the stern, there can be a significant change in the apparent true wind speed during such manoeuvres.

Wind relative direction near 0/360: The age old problem of wind direction near the 0/360 boundary still remains. Since the anemometer is set up with 0/360 at the bow, the relative wind is very often around this heading. Even though the anemometer data are recorded at the data rate generated by the sensor (nominal 1 Hz), there is a problem with the raw data. In particular, when the wind is near 0/360, the TECHSAS files will sometimes contain headings in between, e.g. in the range 150 to 210, reminiscent of when simple numerical averaging of heading was occurring. When these bad headings are used in correct calculation of true wind, bad data are the result.

Irradiance and surface pressure

Downwelling PAR and TIR data are read into both TECHSAS and RVDAS in V rather than W/m2, so factory calibration coefficients should be entered into opt\_*cruise*.m under the mday\_01\_apply\_fcal case.

### 4.1.6 Ocean surface variables: additional processing

Ocean surface variables may come in in multiple streams, e.g. tsg/ocl, sbe38dropkeel, and some are in surfmet along with wind etc.

By default, temperature and salinity are edited for bad times (e.g. when pumps were off), which can be found using mtsg\_findbad.m and added to opt\_*cruise*.m under the mtsg\_medav\_clean\_cal case, and averaged to 30 s using mtsg\_medav\_clean\_cal.m. They may then be calibrated by comparison with bottle samples (salinity) or CTD 5-m temperature. TSG salinity samples are read in at the same time as CTD samples using msal\_01.m (see Section 3). mtsg\_bottle\_compare.m helps compare salinity and choose a time-varying calibration which can be added to opt\_cruise.m under tsgsal\_apply\_cal\*\*\* case. mtsg\_medav\_clean\_cal.m is then rerun with flag usecal set to 1 to incorporate the calibrated data. Finally, mtsg\_surfmet\_merge combines the tsg data with the other surfmet variables.

Temperature variables: On TECHSAS, sea surface temperature (that is, temperature at the seawater intake) is called temp\_r or temp\_m, while the housing temperature (temperature inside the inline CTD, applicable to the conductivity measurements) is called temp\_h. On SCS on the JCR, there are two sea surface temperature sensors, sstemp and sstemp2; the conductivity measurement temperature, tstemp, and the fluorometer temperature, sampletemp. On the Cook, temp\_housing is the temperature applicable to conductivity, temp\_remote that at the seawater intake (actually a little way up the pipe), and sbe35dropkeel\_temp the temperature on the hull.

You can re-run mtsg\_findbad.m, mtsg\_bottle\_compare.m, and m\_tsg\_medav\_clean\_cal.m as many times as required to get a clean record.

If you want to check a calibration already applied, edit the switch at the beginning of mtsg\_bottle\_compare.m from ‘uncal’ to ‘cal’ and rerun.

### 4.1.7 Bathymetry: additional processing

Following ingesting and combining daily data in m\_daily\_proc.m, bathymetry data can be cleaned by interactive script mbathy\_edit\_av, which allows the user to outline and remove bad data points from the EA600 and the EM120/EM122 centre beam for each day.

## 4.2 VMADCP

## 4.2.1 Acquiring and processing/editing data

### 4.2.1.1 VMDAS plus CODAS

Vessel mounted ADCP data processed using the old University of Hawaii CODAS software (Matlab/Python hybrid version, see /local/users/pstar/cruise/sw/uh\_adcp/programs/index.html for documentation) can be loaded into daily and appended Mstar files using mcod\_01.m, mcod\_02.m, and mcod\_mapend.m. The latter sorts by time, so sequences can be added in any order or any number of times.

The full round of processing from VMDAS to Mstar, including the calls to CODAS quick\_adcp.py and gautoedit.m, can be run for a sequence or set of sequences using wrapper scripts vmadcp\_proc.m and vmadcp\_edit.m, as follows:

>> doall = 1; vmadcp\_proc % runs vmadcp\_linkscript to sync and link files; writes q\_py.cnt using initial angle and amplitude set in opt\_cruise.m, and calls quick\_adcp.py to load data into database; calls mcod\_01, mcod\_02, mcod\_mapend

This script will prompt for the instrument to use (75 or 150, probably), sequence number or vector of sequence numbers, and possibly the mode (narrowband or broadband), if not set in opt\_cruise.m

If this fails on “cannot find asetup”, it may indicate there’s not enough data in the sequence

Optional additional processing (better done on multiple sequences, and can be done back at home):

At some point, examine cal/botmtrk/btcaluv.out and cal/watertrk/adcpcal.out (in whichever sequence directories they are found; they will not be generated for every sequence), to refine the angle and amplitude calibrations. Add these to the vmadcp\_proc aa75 and/or aa150 cases in opt\_cruise.m. Once you have done this, you can rerun vmadcp\_proc.m, setting doall = 2:

>> doall = 2; vmadcp\_proc % writes q\_pyrot.cnt using angle and amplitude set in opt\_cruise.m and calls quick\_adcp.py to apply angle and/or amplitude corrections; calls mcod\_01, mcod\_02, mcod\_mapend

>> vmadcp\_edit % writes q\_pyedit.cnt, calls gautoedit.m to enable interactive data editing, calls quick\_adcp.py to apply edits; calls mcod\_01.m, mcod\_02.m, mcod\_mapend.m

### 4.2.1.2 UHDAS plus CODAS

In pycodas terminal (prompt should start with (pycodas) bash-4.2$)

0) If you previously applied calibrations (angle, amplitude, xducer\_dx or \_dy) to data in postprocessing:

rm -rf /local/users/pstar/cruise/data/vmadcp/postprocessing/DY113/proc\_editing/os\*

1) uhdas\_01 #syncs data from acquisition computer to koaeula

2) uhdas\_02 #syncs to postprocessing/proc\_editing (and makes links if necessary)

3) uhdas\_03 #copies previously made edits (archived in proc\_archive) to proc\_editing so they will be applied to newly expanded dataset; adds new data to dataset

4) cd ~/cruise/data/vmadcp/postprocessing/DY113/proc\_editing

5) cd os150nb

6) dataviewer.py -e & #edit using selectors and/or thresholds; remember to apply edits to every segment of time series

7) quick\_adcp.py --steps2rerun apply\_edit:navsteps:calib --auto

8.1) cat cal/watertrk/adcpcal.out

8.2) cat cal/watertrk/guess\_xducerxy.out

8.2) cp -Rp . ../os150nb\_a

9) quick\_adcp.py --steps2rerun apply\_edit:rotate:navsteps:calib --rotate\_amplitude amp --rotate\_angle ang --xducer\_dx dx --xducer\_dy dy --auto #if adcpcal.out shows amplitude or angle calibration needed

10.1) cd ..

10.2) dataviewer.py -c os150nb\_a os150nb & #to see the effect of the edits and calibration. If you are not happy with them, repeat 6)-9). When you are happy:

10.4) rm -rf os150nb\_a #only keep the edited, calibrated version here now

11) cd ../os75nb #and repeat 6)-10)

12) cd ..

13) dataviewer.py -c os75nb os150nb & #to compare two instruments and check for additional needed edits; if any noticed, follow steps above

14) uhdas\_04 #generates .nc files in ~/cruise/data/vmadcp/DY113/postprocessing/proc\_edit/os75nb/contour/ and os150nb/contour/

15) uhdas\_05 #syncs proc\_edit to proc\_archive

## 4.2.2 Output VMADCP station data for LADCP processing

In matlab

16 – 19) vmadcp\_stations\_to\_ladcp(start\_station, end\_station)

16) os = 75; nbb = 1; mvad\_01

os = 150; nbb = 1; mvad\_01

%makes mstar format files containing vmadcp data in /local/users/pstar/cruise/data/vmadcp/mproc/

17) stn = nnn; cast = ‘ctd’; os = 75; nbb=1; mvad\_03

stn = nnn; cast = ‘ctd’; os = 150; nbb=1; mvad\_03

%vmadcp upper ocean velocity profiles corresponding to ctd station nnn; run for each station for which new or edited vmadcp data are available

18) mvad\_for\_ladcp(‘ctd’,nnn,75,1)

mvad\_for\_ladcp(‘ctd’,nnn,150,1)

%makes file of station nnn data to be used as constraint in LADCP processing

19) cfgstr.orient = ‘DLUL’; cfgstr.constraints = {‘GPS’;’BT’;’SADCP’}; process\_cast\_cfgstr(nnn, cfgstr);

# Appendices

## A. Quick guide to using Mexec functions and interacting with Mstar files

You must have ocp\_hydro\_matlab on your Matlab path. Each time you start Matlab:

>> m\_setup % to add paths and set global variables

If you clear the workspace variables:

>> m\_global % to set global variables

To load an Mstar file:

>> [d, h] = mload(filename, ‘/’); % loads all data and header information from filename

Both d and h are structures, d containing “data” and h containing “header” information. The names of the variables in the structure d are also given in h.fldnam, while their units are in h.fldunt. Times in Mstar files (generally in seconds) are relative to the time specified by the vector h.data\_time\_origin [yyyy mm dd HH MM SS] or [yyyy mm dd]. The .nc suffix of filename is not required but the path is.

To save variables from the workspace to an Mstar file:

>> mfsave(filename, d, h) % where d and h are like Mstar data and header structures; see help mfsave for other options

Help mode: on Linux/Unix systems, get\_cropt can be called on the Matlab command line to find out about options:

>> help\_cropt = 1; scriptname = ‘’; oopt = ‘’; get\_cropt

Displays a list of scriptnames and oopts in any of the setdef\_cropt\_\*.m.

>> help\_cropt = 1; scriptname = scriptname; oopt = ‘’; get\_cropt

Displays a list of all calls to get\_cropt in file scriptname.m, if this file exists, and if not, of all files which use scriptname = scriptname to call get\_cropt.

>> help\_cropt = 1; scriptname = scriptname; oopt = oopt; get\_cropt

Displays which of the setdef\_cropt\_\*.m contains this (scriptname, oopt) case (and therefore a help string explaining it), along with a list of opt\_\*.m files which contain it (and therefore examples of modifying the defaults).

\* If a script crashes while writing a file, that file may be left with an "open to write" flag and subsequent calls or scripts will fail. Reset using "mreset". Scripts crashing may also result in a proliferation of temporary wk\*.nc files (which would normally be removed at the end of the script); these contain the date of generation so they can be cleaned up periodically.

\* Once or twice in Mstar scripts we got a mysterious error message along the lines of "not a binary mat file" or "a preference with that name or group already exists". Just re-run the program and next time it will likely be fine; if not, try >> clear all; m\_setup; if that doesn’t work, exit and restart Matlab.

\* Occasionally there may be an error with a library, possibly related to writing the NetCDF files. Restarting Matlab usually clears this up.

Because of the interactive prompting options, sometimes a prompt will appear even if you have passed input arguments in MEXEC\_A.MARGS\_IN. If Matlab says “Busy|”, you don’t need to do anything even if it looks like it is asking for input.

You do not need to be in a particular directory to run the mexec programs, as long as you specify full file paths when using e.g. mload (all the processing scripts are set up to specify full file paths).

\* In matlab/Mstar to open a Mstar nc file: mload

To save data to struct array d, header info to array h:

>> [d h]=mload('filename','/') % ‘/’ means all vars, or you can list the ones you want.

\* >> m\_read\_header(file) allows you to read the header info only

\* >> mhistory: returns to you the text you need to copy into a script to recreate the program steps you just ran (if you have been using MEXEC\_A.MARGS\_IN and not mfsave.m)

## B. Processing checklists

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Cruise: DY146 | CTD number *nnn* |  |  |  |  |
|  | Step | Notes and output |  |  |  |  |
| A | DatCnv.psa  AlignCTD.psa  CellTM.psa | DY146\_CTD\_*nnn*.cnv  DY146\_CTD\_*nnn*\_align.cnv  DY146\_CTD\_*nnn*\_align\_ctm.cnv |  |  |  |  |
| T | ctd\_syncscript | RAW\_CTD\_FILES, ASCII\_FILES |  |  |  |  |
| M | stn = *nnn*; ctd\_all\_part1  mctd\_01  mctd\_02  mctd\_03  mdcs\_01  mout\_1hz\_asc | ctd\_dy146\_*nnn*\_raw.nc  ctd\_dy146\_*nnn*\_raw\_cleaned.nc  ctd\_dy146\_*nnn*\_24hz.nc  ctd\_dy146\_*nnn*\_psal.nc  dcs\_dy146\_*nnn*.nc  ctd.*nnn*.02.asc |  |  |  |  |
| M | stn = *nnn*; mdcs\_03g | Select cast (start and) end  dcs\_dy146\_*nnn*.nc |  |  |  |  |
| M | Edits to opt\_jc238.m: | a) If Niskins leaking or misfired, add flags under mfir\_01 case  b) If cast not full-depth, add water dep in best\_station\_depths case |  |  |  |  |
| M | stn = *nnn*; ctd\_all\_part2  mctd\_04  mfir\_01  mfir\_03  mwin\_01  mwin\_to\_fir  mfir\_to\_sam  station\_summary  mdep\_01 | ctd\_dy146\_*nnn*\_2db.nc  ctd\_dy146\_*nnn*\_2up.nc  fir\_dy146\_*nnn*.nc  win\_dy146\_*nnn*.nc  bot\_dy146\_*nnn*.nc  sam\_dy146\_all.nc |  |  |  |  |
| M | stn = *nnn*; mctd\_checkplots |  |  |  |  |  |
| M | stn = *nnn*; mctd\_rawshow | Look for spikes, bad ranges |  |  |  |  |
| M | stn = *nnn*; mctd\_rawedit  and/or edits to opt\_jc238.m under mctd\_02, rawedit\_auto | If edits required  ctd\_dy146\_*nnn*\_raw\_cleaned.nc |  |  |  |  |
| M | stn = *nnn*; ctd\_all\_postedit  mctd\_02  mctd\_03  mctd\_04  mfir\_03  mfir\_to\_sam  mout\_1hz\_asc | If edits done  ctd\_dy146\_*nnn*\_raw\_cleaned.nc  ctd\_dy146\_*nnn*\_24hz.nc  ctd\_dy146\_*nnn*\_psal.nc  ctd\_dy146\_*nnn*\_2db.nc  ctd\_dy146\_*nnn*\_2up.nc  fir\_dy146\_*nnn*.nc  sam\_dy146\_all.nc  ctd.*nnn*.02.asc |  |  |  |  |
| T | lad\_linkscript\_ix | *nnn*DL000.00, *nnn*UL000.00 |  |  |  |  |
| M | run\_proc\_ladcp (first 3 rounds) | Plots and .mat files |  |  |  |  |

A = on acquisition computer, T = in shell (terminal), M = In Matlab

## D. Known bugs and future changes

### D.1. Bugs

Errors involving MEXEC\_A.MARGS\_OT if a file is interrupted: clear all, run m\_setup, and try again.

Matlab netcdf errors about “a preference with that name or group already exists” occur, apparently when multiple Matlab sessions are accessing the low-level netcdf functions at the same time. The only response is to start again at the point of interruption and hope there won’t be more coincidences, although in extreme cases it may be necessary to restart one or multiple Matlab sessions. The error seems particularly likely to be triggered by running m\_setup, so one adaptation is to keep a couple of Matlab sessions running and set up, rather than starting a new one while Mexec scripts are running in another window . . . however this only reduces rather than eliminates the occurrence of these errors.

Error using netcdflib

Library failure in call to open. eacces:permissionDenied. Error message from the NetCDF

library: "Permission denied".

Error in netcdf.open (line 50)

[varargout{:}] = netcdflib ( 'open', filename, mode );

Error in nc\_attput>nc\_attput\_tmw (line 41)

ncid =netcdf.open(ncfile, nc\_write\_mode );

Error in nc\_attput (line 28)

nc\_attput\_tmw ( ncfile, varname, attribute\_name, attval )

Error in m\_openio (line 20)

nc\_attput(ncfile.name,nc\_global,'openflag','W'); % set to W if file is open to write.

Usual state is R.

Error in mheadr (line 22)

ncfile = m\_openio(ncfile);

Error in mctd\_02a (line 98)

mheadr

Error in ctd\_all\_part1 (line 16)

stn = stnlocal; mctd\_02a; %rename variables following templates/ctd\_renamelist.csv

This means you should remove ctd/ctd\_cruise\_nnn\_raw.nc (or even ctd/ctd\_cruise\_nnn\*.nc) and start again. Make a note on the processing logsheet.

If the beginning part of this error occurs when running smallscript\_botnav, in terminal:

chmod u+w ctd/ctd\_dy113\_nnn\_raw\*.nc

Exit with error because file /local/users/pstar/dy113/mcruise/data/ctd/ctd\_dy113\_002\_2db.nc is already open for write

It may be the case that this program has crashed or been interrupted before, leaving the write flag set in the file

If required you can reset the write flag using

mreset(filename) or mreset(ncfile)

where filename or ncfile.name is a char string containing the name of the mstar file

Error in m\_openot (line 27)

ncfile = m\_exitifopen(ncfile); % exit if write flag set

Error in mcalc (line 57)

ncfile\_ot = m\_openot(ncfile\_ot);

Error in mctd\_04 (line 176)

mcalc

>> mreset('ctd/ctd\_dy113\_002\_2db.nc')

About to reset mstar openflag on file ctd/ctd\_dy113\_002\_2db.nc.

Do you really want to do this ?

Reply y/yes. Default is no. y

File ctd/ctd\_dy113\_002\_2db.nc has been modified

Attempt to reference field of non-structure array.

Error in mtposinfo (line 51)

tin = pdata.time+MEXEC\_G.uway\_torg;

Error in mctd\_02a (line 111)

[botlat botlon] = mtposinfo(tbotmat);

Error in ctd\_all\_part1 (line 16)

stn = stnlocal; mctd\_02a; %rename variables following templates/ctd\_renamelist.csv

run techsas\_linkscript in terminal (have to do this each new day UTC)

some error about f.ladcpdo

run lad\_linkscript\_ix and make sure LADCP files were copied/links were made

Notes: If you run m\_setup.m with skipunderway set to 0 (set in opt\_da001.m) before mrtables\_from\_json.m is available, it will get stuck. If one of the matlab scripts seems to freeze and not respond to ctrl-C, try hitting enter; if this unsticks it (but it fails), it may be getting stuck on the system command where it tries to run the psql string; check ~/.pgpass, and if that doesn’t fix it, try putting a debug point in mr\_try\_psql.m to check the psql string and maybe try running it from a terminal directly.