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

Contents

1. Introduction

1.1 About this guide

1.2 What is Mexec?

1.3 Caveats, changes and bugs

2. Setting up a new cruise

3. CTD and Niskin bottle sample data

3.1 SeaBird data acquisition and processing

3.2 Mexec data processing

3.3 Water bottle sample data

3.4 Sensor calibration in Mexec

3.5 Outputting data in other formats

4. Underway data: navigation, surface and bathymetry data

4.1 TECHSAS/SCS data access

4.2 Preparation at the start of the cruise

4.3 Mexec data processing

Appendices

A. A bit more detail on Mexec functions

B. Handy hints and tips

C. List of cruise-specific options

D. Known bugs and planned changes

­**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. Script/function names are in bold, except where they are part of examples; variables which 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*).

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 \*\*\*.

This guide was first written by Penny Holliday. It 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 shells 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 Mexec system and code was developed over a number of years by scientists at the UK National Oceanography Centre, Southampton, principally Brian King, with contributions from numerous others including G. Evans, Y. Firing, C. Florindo Lopez, E. Kent, G. McCarthy, B. Moat, A. Sanchez-Franks and D. Smeed.

The code has undergone numerous revisions, some preserved in git and some only in each cruise’s end-of-cruise processing workstation backup. Major organizational 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 frozen]: git.noc.ac.uk/MEXEC/mexec\_processing\_scripts, git.noc.ac.uk/MEXEC/mexec, and git.noc.ac.uk/MEXEC/mexec\_exec).

JC191 and DY113 (2019/20): revision of netcdf file-interface code to use Matlab-native functions (replacing bespoke functions developed before these 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): version 4: merging of mexec and mexec\_processing\_scripts along with other related Matlab code into a single git repository.

Moderate attempts at backwards compatibility have been made, with the goal that newer code could be used to reprocess older cruises (without starting completely from scratch). Options files for pre-JR15003 cruises were added and generally kept up to date with changes to code through DY113, but 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. 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 most variable names (but not at every stage?\*\*\*).

The current status is that Mexec consists of (components of) of two git repositories, git.noc.ac.uk/OCP/ocp\_hydro\_matlab (containing all the Matlab code formerly in MEXEC/mexec and MEXEC/mexec\_processing\_scripts, along with other programs that use some of the same code), and git.noc.ac.uk/OCP/mexec\_exec (containing shell scripts, formerly in MEXEC/mexec\_exec).

**1.2.1 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/.

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 that (may) change from cruise to cruise**

Program version control in Mexec v3 and v4 operates by way of a set of scripts setting default and cruise-specific options: each cruise requires an **opt\_*cruise*.m** (e.g. **opt\_dy113.m**) in mexec\_processing\_scripts/cruise\_options/. Parameters and variables that might need to change from cruise to cruise (such as which CTD is primary, the calibration functions for conductivity and other parameters, and more) are set by processing scripts calling **get\_cropt.m**, which

1. Calls the **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 scripts) check settings and warn about invalid or unset options.

A list of scripts and their options is given in Appendix C\*\*\*.

The defaults are split, purely for convenience, across four scripts: **setdef\_cropt\_cast.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 scripts contain explanations for each set of options and therefore may be useful to examine (see also Help mode, below), but general users should not edit them. Instead, make changes to **opt\_*cruise*.m** for your cruise (and raise an issue if you think the defaults themselves should change). Do not duplicate (unchanged) default settings in **opt\_*cruise*.m**.

Help mode: on Linux/Unix systems, get\_cropt can be used 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).

Information obtained or modified during the course of Mexec processing, but which pertains to the cruise data rather than a particular choice for processing, may be contained in either **opt\_*cruise*.m** or in a separate text file specified in one of the **opt\_*cruise*.m** cases. For instance, a list of salinity sample flags to modify might be set in the msal\_01 case of **opt\_*cruise*.m**, or read in from a list by **msam\_02b.m**. \*\*\* 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.

**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.m** 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 \*\*\*).

**1.2.4 Mstar file format and quick guide to interacting with existing Mstar files**

.

You must have mexec\_processing\_scripts 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\_common % 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

**1.2.5 Directory structure and processing locations**

The data processing directory and subdirectories are specified in m\_setup.m or by m\_setudirs.m (put this later?); under this, the scripts either expect or will create:

If you run mexec\_exec/cruisedir\_setup to set up your processing directory structure (see Section 2), symbolic links are relative, so that processing can be redone or continued in another directory structure (for instance, on a network drive ashore) by copying the cruise processing directory over.

**1.3 Changes and bugs**

To submit edits, report bugs or suggest changes, raise a pull request or issue at git.noc.ac.uk/OCP/ocp\_hydro\_matlab or git.noc.ac.uk/OCP/mexec\_exec, or email [yvonne.firing@noc.ac.uk](mailto:yvonne.firing@noc.ac.uk). (Please see the lists of known bugs and planned additions in Appendix D first.)

We would like to record cruise-specific options used as well as integrate user improvements, so we encourage you to create a pull request or send us your scripts at the end of your cruise. To make it easier for us to track and integrate changes, please put any new scripts/functions that you think might be added to and maintained in the code base into ocp\_hydro\_matlab/ (and, ideally, in the issue or email include some context about your changes and/or additions), and keep any code that is working code just for your cruise (e.g. quick/specific code making plots of data) in a separate directory (i.e. outside ocp\_hydro\_matlab).

**2. Setting up a new cruise**

i) Clone or download two git repositories (master or the latest cruise branch):

git.noc.ac.uk/OCP/ocp\_hydro\_matlab

git.noc.ac.uk/MEXEC/mexec\_exec

Add mexec\_exec and its subdirectories to your shell path (in .bashrc, .zshrc, .tcshrc, etc.). Add ocp\_hydro\_matlab and its subdirectories to your Matlab startup path.

ii) Edit the cruise name and processing base directory (examples in this guide will assume processing will be done in ~/*cruise*/mcruise/ [e.g. /local/users/pstar/dy113/mcruise/]) in 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.

iii) 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. \*\*\*

iv) Edit shell scripts in mexec\_exec/: ctd\_syncscript, lad\_syncscript, techsas\_linkscript\*\*\*, uhdas\_00\_\*\*\*.

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 edit and use mexec\_exec/cruise\_backup to sync the processing directory and scripts to an external hard drive (i.e. regularly by adding a call to cruise\_backup to crontab).

v) Edit template files in ocp\_hydro\_matlab/mexec\_processing\_scripts/varlists/ \*\*\*deprecated?

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 put in place by the conf\_script 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.

vii) Edit ocp\_hydro\_matlab/mexec\_processing\_scripts/**m\_setup.m**.

Most things you should or may need to modify are near the top of the file, including cruise name, and year of the data time origin. Two flags, “quiet” and “ssd”, determine how much information will be displayed to the screen. Ladcp\*\*\* variable names or actual lines \*\*\*

viii) If necessary, edit:

a) 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. \*\*\*mrvdas

b) ocp\_hydro\_matlab/mexec\_processing\_scripts/underway/**m\_setudir.m**: if you added new Mexec stream abbreviations in a), 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.

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

c) 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.

ix) Generate an empty cruise-specific options file, ocp\_hydro\_matlab/mexec\_processing\_scripts/cruise\_options/**opt\_*cruise*.m**, for your cruise. See below, as well as other files in that directory, for examples.

**3. CTD data and Niskin 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

**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, by running the following three steps:

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 may include oxygen hysteresis correction using SBE default parameters, but it is recommended not to apply the correction here but instead to apply 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 to change the dooxyhyst flag in the mctd\_02 case of **opt\_*cruise*.m**).

The output file names should contain the three-digit sequential station/cast number (ideally, something like ctd\_*cruise*\_*nnn*.cnv).

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*\_\*.nc contain CTD time series or profiles, with different stages of processing/editing/averaging/adding derived variables

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 CTD data

sam\_*cruise*\_all.nc contains CTD data from bottle firing times along with corresponding sample data (for all the stations)

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

An example checklist of common processing steps is given in Table \*\*\*.

The basic steps for CTD processing following a cast in the dy146 branch are:

$ ctd\_syncscript

Copies .hex, .cnv, \_align\_ctm.cnv, and .bl files from SM computer to koaekea.

>> stn = {N}; ctd\_all\_part1

ctd\_all\_part1.m calls:

1. mctd\_01.m loads and (new) renames variables and saves in ctd\_{cruise}\_{nnn}\_raw.nc.
2. mctd\_02.m does conversions, edits, and corrections, calling (new) ctd\_apply\_autoedits.m, ctd\_apply\_oxyhyst.m, select\_calibrations.m, and apply\_calibrations.m to produce \_raw\_cleaned.nc and \_24hz.nc files. The syntax for specifying edits, corrections, and calibrations in setdef\_cropt\_ctd.m and opt\_{cruise}.m was also modified, allowing (among other things) ctd\_apply\_oxyhyst to apply different corrections for different sensors.
3. mctd\_03.m selects primary sensors, computes derived variables and averages to 1 Hz, using (new) grid\_profile.m, producing \_psal.nc. Note that grid\_profile.m replaces a variety of averaging/interpolating/filling behaviour using (old) merge\_avmed.m, mintrp2.m, and other code in previous versions, calling (new) gp\_smooth.m, gp\_fillgaps.m, gp\_binav.m, and gp\_intav.m.
4. 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.

>> stn = {N}; ctd\_all\_part2

ctd\_all\_part2.m calls:

* 1. mctd\_04.m separates down and up casts, optionally applies m\_loopedit.m to the downcast data, and averages to 2 dbar, producing \_2db.nc and \_2up.nc.
  2. mfir\_01.m gets times and scans of Niskin bottle firing from .bl file and puts in fir\_{cruise}\_{nnn}.nc.
  3. mfir\_03.m gets CTD data from these scans from the 1-Hz \_psal.nc file and adds to fir\_{cruise}\_{nnn}.nc.
  4. mwin\_01.m gets winch information from the underway stream and saves in win\_{cruise}\_{nnn}.nc.
  5. mwin\_to\_fir.m adds it to fir\_{cruise}\_{nnn}.nc.
  6. mfir\_to\_sam.m puts bottle firing data from fir\_{cruise}\_{nnn}.nc into appended sam\_{cruise}\_all.nc.
  7. 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, and if they cannot be calculated by CTD+altimeter are read from opt\_{cruise}.m (which should be filled in for non full-depth casts).
  8. mdep\_01.m adds the depths to the various ctd\_{cruise}\_{nnn}\_\*.nc files.

>> stn = {n}; mctd\_checkplots

mctd\_checkplots.m produces a set of plots to check for sensor drift or other problems.

>> stn = {n}; mctd\_rawshow

mctd\_rawshow.m allows inspection of 24 Hz data. If this reveals editing needed, there are 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

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 propagate changes to the \_raw\_cleaned.nc file through to other files.

Shell script **ctd\_syncscript** can be customised to copy the .cnv files from the network data drive where they have been made available by the CTD operator (whose location on the data processing workstation may vary) to the Mexec directory. The opt\_cruise.m file is then edited to add the naming convention of the files if it differs from the default \*\*\*.

When starting Matlab for the first time, run **m\_setup.m** 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.m** to regenerate the global variables.

The MSTAR processing is split into several phases, some of which are grouped in wrapper scripts. A typical sequence for processing CTD data following a cast is as follows.

>> stn = *nnn*; ctd\_all\_part1

**ctd\_all\_part1.m** calls the following:

**mctd\_01.m** reads in 24Hz CTD data from the .cnv file and changes variable names and file format, but not values, except for the cast position in the file header, including the operator-input position from the header, renames SeaBird variable names based on varlists/ctd\_*cruise*\_renamelist.csv, updates the header positions based the CTD NMEA variables, if available, or on the underway techsas or scs navigation data, , into ctd\_*cruise*\_*nnn*\_raw.nc, including the operator-input position from the .cnv file header, renames

**mctd\_02.m** changes variable values by applying corrections to raw data as specified in opt\_cruise.m:

**automatic edits: ctd\_apply\_autoedits.m: scan, range, pumps, and despiker**

**saved to ctd\_cruise\_nnn\_raw\_cleaned.nc**

**(optionally) celltm\_align**

**Oxygen hysteresis correction and any other conversions**

**Calibration**

saved to ctd\_*cruise*\_*nnn*\_24hz.nc:

by default, raw data are corrected for oxygen hysteresis using the SBE parameters under the assumption that these have **not** been applied in SBE processing; see the cruise options files for other options including reversing oxygen hysteresis correction, converting turbidity volts to turbidity, and reversing or applying cell thermal mass correction\*\*\*

**mctd\_03.m copies variables from primary sensor (set in opt\_cruise.m), calculates salinity and other derived variables and** averages data to 1 Hz to save in ctd\_*cruise*\_*nnn*\_psal.nc

**mdcs\_01.m** selects the start and bottom of the good data of the cast, saved in dcs\_cruise\_nnn.nc

**if ladcp: mout\_1hzasc**(*nnn*) generates an ascii listing of the 1hz (ctd\_*cruise*\_*nnn*\_psal.nc) file ready for use in the LADCP processing. Each file, ctd\_*cruise*\_*nnn*\_1hz\_txt, is saved in data/ladcp/ix/data/CTD.

>> stn = *nnn*; mdcs\_03g

**mdcs\_03g.m** allows the user to decide which scan numbers mark the start of the downcast and the end of the upcast. This is a graphical interface. 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). The start and end scans selected in the GUI are written to dcs\_*cruise*\_*nnn*.nc.

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

**ctd\_all\_part2.m** calls the following:

**mctd\_04.m** extracts downcast and upcast data from the psal file using index information from the dcs file, and sorts, interpolates over gaps, and averages to 2db (output to ctd\_*cruise*\_*nnn*\_2db.nc and ctd\_*cruise*\_*nnn*\_2up.nc);

**mfir\_01.m** reads in information from SeaBird .bl file and creates netCDF fir file;

**mfir\_02.m** merges time from ctd file onto fir file using scan number (output to fir\_*cruise*\_*nnn*\_time.nc);

**mfir\_03.m** merges CTD upcast data onto fir file;

**mfir\_04.m** pastes CTD fir data into sam\_*cruise*\_*nnn*.nc;

**mwin\_01.m** creates win file to hold winch data and extracts times from start and end of ctd\_*cruise*\_*nnn*\_1hz.nc;

**mwin\_03.m** merge winch wire out data onto fir file;

**mwin\_04.m** paste winch fir data into sam file;

**mbot\_00.m** inserts default Niskin bottle numbers and firing flags into bot\_*cruise*\_*nnn*.csv, calling **get\_cropt.m** to modify niskin-position correspondences;

**mbot\_01.m** writes these to bot\_*cruise*\_*nnn*.nc, and calls **get\_cropt.m** to modify selected flags

**mbot\_02.m** copies bot\_*cruise*\_*nnn*.nc to sam\_*cruise*\_*nnn*.nc

At this point the data can be examined using some scripts to generate standard plots:

>> stn = *nnn*; mctd\_checkplots % and answer 0, 1, or 2 (more than this is likely to be illegible) to the query about number of previous stations to plot with station *nnn*

**mctd\_checkplots.m** generates a series of plots of raw, 1hz and 2db data, and allows a series of casts to be plotted together. \*\*\*what to look for in each of the plots\*\*\*

>> stn = *nnn*; mctd\_rawshow;

**mctd\_rawshow.m** generates plots of raw and 1 hz data, which should be examined for data quality. If bad data are found, run mctd\_rawedit; if not, skip this step.

>> stn = *nnn*; mctd\_rawedit % and follow prompts

In some cases mctd\_rawshow may indicate that it is necessary to make corrections before cellTM\*\*\*

**mctd\_rawedit.m** is a graphical interface thatallows the user to manually select bad data cycles in temp, cond and oxygen. Preserves original raw file as ctd\_*cruise*\_*nnn*\_raw\_original.nc and outputs new file ctd\_*cruise*\_*nnn*\_raw\_cleaned.nc. The cleaned file is linked to by a new symbolic link called ctd\_*cruise*\_*nnn*\_raw.nc so that **mctd\_02b.m** and following scripts work on the cleaned version if it exists.

The edited scans from each field are recorded in ctd/mplxyed\_*yyyymmdd*\_*hhmmss*\_ctd\_*cruise*\_*nnn*.

When there are repeated or extended faults in the CTD communications, or persistent contamination in the intake, it may make sense to do some pre-editing before the graphical editing. The mctd\_rawedit case in opt\_cruise.m can be set to edit out bad data over a range of scans and/or to edit out data outside of specified ranges, applying these edits before producing the plots for the GUI.

It may also be necessary to edit earlier on \*\*\*

>> klist = *nnn*; smallscript\_postedit

The editing is done on the raw data file, so after the edits are finished, the derived files must be re-generated, by running **mctd\_02b.m**, **mctd\_03.m**, **mctd\_04.m**, **mfir\_03.m**, and **mfir\_04.m**, called by **smallscript\_postedit.m**, which can be run on a single file (as above) or a group of files (klist can be a vector; if not set klist will be taken from the smallscript case of **opt\_*cruise*.m**).

**\*\*\*** **At the end of this section (i.e. 3.2.3), or maybe just at the end of each wrapper script, I would add something on what to do or how to edit if you've made a mistake somewhere. E.g. does a file need to be deleted, or from what level can things simply be re-run (knowing it will overwrite the mistake, but not some other vital info), etc...\*\*\***

**3.2.3 Processing steps requiring information from other instruments**

Position and depth information can be added to the Mstar ctd files from the underway navigation and a file called station\_depth\_*cruise*.mat. \*\*\*where does this fit in with the steps in 3.2.2?\*\*\*

**populate\_station\_depths.m** produces and updates station\_depth\_*cruise*.mat.

It can read in depths from an ascii file, or from LDEO processed LADCP files, or estimate them from CTD bottom depths and altimeter readings; the method to use is set in **opt\_*cruise*.m**, where depths for selected stations can also be explicitly set. **populate\_station\_depths.m** will add to station\_depth\_*cruise*.mat depths for all stations for which they are available (using the method specified in **opt\_*cruise*.m**), so it does not necessarily need to be run once per station; however, it should be up to date for a given station before running smallscript\_botnav (below) for that station.

>> smallscript\_botnav

**smallscript\_botnav.m** takes in (for a single file or a group of files) the depth, navigation, and bottle data and runs the following:

**mbot\_01.m** takes bottle firing quality flags manually set in bot\_*cruise*\_01.csv. Output: bot\_*cruise*\_*nnn*.nc.

**mbot\_02.m** pastes the bottle firing codes into sam\_*cruise*\_*nnn*.nc

**mdep\_01.m** reads water depths from station\_depth\_*cruise*.mat (generated by **populate\_station\_depths.m**), and pastes this information into headers of all CTD files.

**mdcs\_04.m** takes the lat and lon from the navigation (pos\_*cruise*\_01.nc, generated by the daily processing of underway datastreams) at the time of start, bottom and end of each cast and pastes into dcs\_*cruise*\_*nnn*\_pos.nc.

**mdcs\_05.m** pastes the lat and lon for the bottom of the cast into the headers of all CTD files.

**mout\_sam\_csv**

**msam\_02b**

**mctd\_02a**

**mctd\_02b**

**mctd\_rawedit**

mctd\_makelists

msam\_checkbottles\_01

msam\_checkbottles\_02

msec\_run\_mgridp

more about how history and versions work, how to keep history from getting excessively long (?) (e.g. if you really start all aspects of a cast’s processing fresh from the sbe files, delete the history file first)

proliferation of wk files

**3.3 Water Bottle Sample Data**

The aim of the sample data processing for CTD profiles 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 the sam\_*cruise*\_*nnn*.nc files during running of **ctd\_all\_part2.m**, as described above. This section describes how the water sample data are included in the process. All sample data must first be saved in ascii csv files; Mexec scripts read the ascii files and create Mstar files for each sample type (the '\_01' scripts), and paste data from these into sam\_*cruise*\_*nnn*.nc (the '\_02' scripts). **msam\_apend.m** then concatenates the profile files into sam\_*cruise*\_all.nc.

Underway samples are treated in a similar way, and saved in tsg\_*cruise*\_all.nc (see Section 4.1.6).

The key decisions at the start of the process involve settling on a consistent and suitable format for the ascii files. Information logged on a given type of sample should be put into a csv file to be read in using Matlab’s dataset utility. The order of columns does not matter, but certain column headers are required, as described in the help for **msal\_standardise\_avg.m** and **moxy\_01.m**. Absent data should be -999, while flag values for bottles not sampled should be 9. Sample data should be flagged according to WOCE standard flags given in the GO-SHIP Repeat Hydrography manual (<http://www.go-ship.org/HydroMan.html>). Another file, bot\_*cruise*\_01.csv, should be constructed to give quality flags for the Niskin bottles themselves (see Section 2 point xi).

• **Salinity:** ascii concatenated comma-separated-value file sal\_*cruise*\_01.csv. The spreadsheets from each salinometer run should have header information corresponding to the run. The data values are then concatenated into sal\_cruise\_01.csv, removing from this file the header information except for the single (first) row of column headers, which become database field names. See help for **msal\_01.m** and **msal\_standardise\_avg.m** for examples of acceptable formats, ways to number samples and indicate standard sea water samples, etc.

• **Oxygen:** ascii comma-separated-value files oxy\_*cruise*\_*nnn*.csv The files should be prepared with header lines and a range of columns of data. See **moxy\_01.m**.

• **Nutrients:** ascii comma-separated-value files nut\_*cruise*\_*nnn*.csv. The files should be prepared with header lines and a range of columns of data. See \*\*\*

Carbon

CFCs

smallscript\_load\_botcaldata

flags: niskin suspicious but not sure it did leak: initial flags should be 2 (good), 3 (leaking [and did not sample]), 7 (unknown problem [maybe leaking but did sample]), 9 (did not sample)? Then after examining data update flags to 2 (good), 3 (leaking), 4 (did not trip correctly), 9 (did not sample)?

**3.4. Sensor Calibration in Mexec**

The steps below are described in the simplest order, assuming temperature, conductivity, and oxygen are all being calibrated. However, steps can be skipped or run out of order (for instance, oxygen values could be compared before calibrating temperature and salinity, as long as the associated uncertainty is kept in mind). These scripts can be run on all stations, or on a specified list of stations if calibration data for all are not available, or if only some stations have changed flags. Keep in mind, however, that the goal is to calibrate each sensor, not each cast; comparisons may not be very useful until sufficient data are accumulated.

3.4.1 steps that ingest and evaluate calibration data

>> caldata\_all\_part1 % optionally first set klist to a list of stations; otherwise uses klist from opt\_*cruise* smallscript case

**caldata\_all\_part1.m** puts temperature and salinity calibration data in the master sample file by calling the following scripts for a list of stations: (\*\*\*msbe35 only if directory found\*\*\*)

**msbe35\_01.m** reads in the SBE35 ascii files listed in lsbe (this file can be generated by listing on the command line). Cruise-specific options can be used to set flags for quality of reading (for instance, if a bottle were fired on the fly, the SBE35 reading would be questionable or bad).

**msbe35\_02.m** pastes them into sam\_*cruise*\_*nnn*.nc

**msal\_01.m** reads the comma-delimited concatenated bottle salinity file sal\_cruise\_01.csv into matlab and saves data from station *nnn* as sal\_*cruise*\_*nnn*.nc. Cruise-specific options allow the salinometer bath temperature and conductivity ratio offset to be set, if they are not included in the csv file. **msal\_01.m** calls:

**msal\_standardise\_avg.m**. This function will compute offsets, if not supplied, and optionally produce plots of different readings to allow bad readings to be excluded or bad samples to be flagged by editing **opt\_*cruise*.m**.

**msal\_02.m** pastes the bottle salinity into sam\_*cruise*\_*nnn*.nc

**msam\_02.m** computes residuals

**msam\_apend.m** concatenates the station sample files into sam\_*cruise*\_all.nc.

>> sensname = ‘temp’; ctd\_evaluate\_sensors % use the plots and fits produced to choose a calibration for temperature sensors and edit the temp\_apply\_cal case of **opt\_*cruise*.m**

**ctd\_evaluate\_sensors.m** compares data from the CTD to the calibration sample data. The quantity to be compared is set by variable sensname. The script generates plots of residuals against time and pressure to allow the user to get a sense of how the sensors are behaving, and to determine an appropriate calibration function to enter in to the temp\_apply\_cal (for temperature), cond\_apply\_cal (for conductivity), or oxy\_apply\_cal (for oxygen) cases of **opt\_*cruise*.m**.

It allows data to be examined in groups of primary or secondary sensors, including multiple such groupings (if a sensor was changed during the cruise, or there was an apparent calibration shift at any point). The groups of sensors must be added to the ctd\_evaluate\_sensors case of **opt\_*cruise*.m**.

The comparisons produced by **ctd\_evaluate\_sensors.m** may also indicate bad or questionable bottle samples (or just questionable comparisons, for instance in regions of high gradient), which can be flagged as 4 or 3 respectively by editing the msbe35\_01 (for temperature), msal\_01 (for salinity/conductivity), and moxy\_01 (for oxygen) cases of **opt\_*cruise*.m**.

msam\_checkbottles\_01

msam\_checkbottles\_02

\*\*\*show examples of each?\*\*\*

>> precalt = 1; sensname = ‘cond’; ctd\_evaluate\_sensors % and edit cond\_apply\_cal case of **opt\_*cruise*.m**

Calibrations entered into the temp\_apply\_cal, cond\_apply\_cal, or oxy\_apply\_cal cases of **opt\_*cruise*.m** can be applied to any data by calling temp\_apply\_cal etc.\*\*\* to be tested (before being applied to the Mstar files) by setting precalt, precalc, or precalo, respectively, to 1 before running **ctd\_evaluate\_sensors.m**. This will produce the same set of plots comparing the calibration data with the calibrated CTD data. this lets you apply the calibration to data without modifying the files, so you can test it.

3.4.2 steps that apply calibrations to CTD data

>> smallscript\_tccal

**smallscript\_tccal.m** applies the temperature and conductivity calibrations set in the temp\_apply\_cal and cond\_apply\_cal cases of **opt\_*cruise*.m** to the Mstar files by calling:

**mctd\_tempcal.m** and **mctd\_condcal.m** for both sensors to apply the calibrations to ctd\_cruise\_nnn\_24hz.nc

**mctd\_03.m** and subsequent files to propagate the calibrated data into the other Mstar files

If bottle sample flags were changed in **opt\_*cruise*.m** based on the results of **ctd\_evaluate\_sensors.m**, **msal\_01.m** and **msal\_02.m** should also be run, by uncommenting them in **smallscript\_tccal.m**.

>> caldata\_all\_part2

**caldata\_all\_part2.m** calls

**moxy\_01.m** to read the ascii file into matlab and saves as oxy\_*cruise*\_*nnn*.nc

**moxy\_02.m** pastes the bottle oxygens into sam\_*cruise*\_*nnn*.nc.

**msam\_oxykg.m** calculates bottle oxygen in units of umol/kg using CTD salinity and bottle oxygen fixing temperature. Output variables: botoxysams and botoxynoc sam\_*cruise*\_*nnn*.nc.

**mnut\_01** reads the ascii file into matlab and saves as nut\_*cruise*\_*nnn*.nc

**mnut\_02** pastes the bottle data into sam\_*cruise*\_*nnn*.nc

**mnut\_03** computes organic from total and inorganic nutrient values in sam\_*cruise*\_*nnn*.nc

\*\*\*similarly for co2, cfcs, ch4, as appropriate\*\*\*

**msam\_apend** concatenates the station sample files into sam\_*cruise*\_all.nc.

>> sensname = ‘oxy’; ctd\_evaluate\_sensors % and edit oxy\_apply\_cal case of **opt\_*cruise*.m**

It is ideal to evaluate the oxygen calibration after conductivity and temperature calibrations have been applied, since oxygen concentration depends on density.

>> smallscript\_ocal

**smallscript\_ocal.m** does the equivalent of **smallscript\_tccal.m** for oxygen

At this point, the data in the 24hz, 1hz, psal, 2db, 2up, and sam files are all calibrated.

bottle\_data\_flags.txt and msam\_02b can be used to update flags in the sample file to make editing and inspection easier, but at some point you should move the flags from bottle\_data\_flags.txt to the msal\_01, moxy\_01, mco2\_01, mnut\_01, mcfc\_01 cases in opt\_cruise, and rerun those respective scripts for the necessary stations so that the flags are applied in the sal\_, oxy\_, etc. files (this is so that when those are updated and transferred to sam\_ files, msam\_02b doesn’t have to reapply them to the sam\_ files)

**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);

And if you have a file\*\*\* of SADCP data for the station, …

**3.5.3 WOCE exchange format CTD and bottle data**

**mout\_cchdo\_sam.m** and **mout\_cchdo\_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\_cchdo\_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 which are not.

**3.5.3 Summary tables**

**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**

**m\_daily\_proc**

**later: mtsg\_findbad, mtsg\_medav\_clean\_cal, mtsg\_bottle\_compare, mtsg\_medav\_clean\_cal, mtsg\_surfmet\_merge**

**4.1.1 Data access**

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

The following Mexec Matlab commands can be used for a quick look at TECHSAS data; substitute ms for mt for corresponding SCS commands.

help mtechsas lists the 'mt' commands

mtlookd tells you filename, start, end

mtlookdf faster version that doesnt count datacycles

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

mtdfinfo winch provides info about that datastream (eg winch)

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

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

help mtlistit for how to use mtlistit to list segments of data

**4.1.2 Preparation at the start of the cruise**

**m\_setudir.m**, called by **m\_setup.m**, creates the directories in which Mstar versions of the underway data will be placed. If the ship does not record a certain data stream (eg SBE35 not used on JC) that short name is ignored by the mexec scripts. But if a processed data directory is not present, the scripts will also ignore its corresponding stream, and the data will not be processed. Therefore you can exclude certain streams from Mexec processing (if they are unimportant, or low quality) by editing **m\_setudir.m** at the beginning of the cruise (see Section 2).

**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 = [*nnn*]; m\_daily\_proc

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

**m\_daily\_proc.m** goes through the list of underway data streams found in mtnames (for techsas) or msnames (for scs), finds which ones are present in 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/sim/sim\_*cruise*\_d*nnn*\_raw.nc).

It then performs additional processing and cleaning steps on some streams by calling **mday\_01\_clean\_av.m**, which has cases for different streams. The automatic processing includes renaming variables to standard names (e.g. head\_gyr, depth) 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*nnn*\_edt.nc.

For bathymetry, when both the Simrad EA600 (sim) and the swath EM122 centre beam (em120) streams are present, **msim\_02.m** and **mem120\_02.m** are called to paste in the depths from the other instrument for subsequent comparison.

The final daily automatic processing step is to call **mday\_02.m,** which appends the daily file to create a master cruise file for each data stream (eg sim\_*cruise*\_01.nc). The list of daily files appended into the master files is given in the header information of that file.

After all daily steps have been run, **mbest\_all.m**, **mtruew\_01.m**, **mtsg\_medav\_clean\_cal.m**, and (for scs) **upate\_allmat.m** are called to produce further combined/averaged files.

Note: if daily processing is run more than once for an individual day, the master file will have the day's data appended again and may need to be recreated. It may be useful for future cruises to run the appending steps in a separate script.

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**

Bestnav: **mbest\_all.m** runs a series of scripts to produce the master bestnav file, bst\_*cruise*\_01.nc. The streams used for best position and heading are set in **m\_setup.m**. Scripts merge heading and position so that there is a complete file containing position, heading, course and speed made good, and distance run. The data are reduced to a 30-second time base and heading is properly vector averaged. This is the ‘definitive’ cruise navigation file. In order to avoid the problem of housekeeping variables across daily files, the bestnav processing is rerun from the start of the cruise each time it is required. There is therefore only ever one bst\_*cruise*\_01.nc file.

**4.1.5 Meteorology: additional processing**

Wind variables: Ship speed, position and heading from the bst navigation file are merged onto the wind data in the surfmet. The absolute wind speed is calculated and vector averaged in one multi-step script **mtruew\_01.m**. As with bst processing, this is rerun for the entire cruise each time the data are updated. The output files from this processing are  
met\_*cruise*\_true.nc  
met\_*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, eg 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 found in the surflight stream, which also contains barometer pressure. These streams were ingested and stored, but no further processing was undertaken.

**4.1.6 Ocean surface variables: additional processing**

Salinity is the only variable that is calibrated.

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.

The appended data can be further processed as follows. These steps can be carried out at any time during the cruise, but need to be run a final time at the end since they act on the appended file.

i) run **mtsg\_medav\_clean\_cal.m** to average the appended file

ii) run **mtsg\_findbad.m** to find limits of times when the data were bad (likely when the pumps where switched off) by selecting them on a graph

iii) edit the mtsg\_cleanup case in **opt\_*cruise*.m** to hardwire in the selected bad time ranges (displayed to the screen at the end of mtsg\_findbad) as well as limits for different variables.

iv) run **mtsg\_medav\_clean\_cal.m** to apply the bad time limits (calling mtsg\_cleanup.m). The default is to NaN all variables within the time ranges set in **opt\_*cruise*.m**, unless **opt\_*cruise*.m** contains code excluding some variables.

v) run **mtsg\_01.m** to load TSG bottle sample salinity data from the concatenated salinity csv file (see Section 3.4) to an Mstar-format file.

vi) run **mtsg\_bottle\_compare.m** to plot bottle and TSG salinities together; determine a constant or simple time-dependent offset to bring them into alignment, or just use the smoothed difference, computed by calling **filter\_bak.m** and saved to tsg\_smdiff.txt.

vii) edit the tsgsal\_apply\_cal case in **opt\_*cruise*.m** with calibration determined above, or to interpolate the smoothed differences computed by **mtsg\_bottle\_compare.m** to the TSG times and use that series as the offset.

viii) run **mtsg\_medav\_clean\_cal.m** to apply calibration

You can re-run **mtsg\_findbad.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 **m\_daily\_proc.m**, bathymetry data can be cleaned by interactive scripts **msim\_plot** and **mem120\_plot**, which allow the user to select bad data points from the EA600 and the EM120/EM122 centre beam for each day. To incorporate the cleaned data into the appended files at the end of the cruise, remove sim\_*cruise*\_01.nc and em120\_*cruise*\_01.nc and rerun **mday\_02.m** for all days for these two streams (see **m\_daily\_proc.m** for syntax).

**4.2 VMADCP**

4.2.1 Old 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

4.2.2 Python 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

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);

can’t process cast 79 using os150 ‘ctd’ method because there is too little good data (too short a time on station)

Making plots:

new:

use sectinfotest.txt as a model (transposed format, includes section names)

quick\_plots.py --inst os150nb --sifile sectinfotest.txt --sect cb

and similarly for the other sections listed in the sectinfo file (you’ll have to run the command more times as it will only make plots for one section each time)

Appendices

**A. A bit more detail about Mexec functions**

The Mexec functions found in subdirectories of mexec\_v3/source/ can either interactively query for inputs, or take inputs from global cell array MEXEC\_A.MARGS\_IN (or both, if MEXEC\_A.MARGS\_IN has fewer elements than the function is expecting). Future versions may evolve towards more standard argument parsing, without the querying mode.

More detail: MEXEC\_A having been declared a global variable by m\_setup, it is available within any function by calling m\_common. Within each top-level function (e.g. msave, mcalib), MEXEC\_A.MARGS\_IN is copied to MEXEC\_A.MARGS\_IN\_LOCAL and cleared. Then m\_getinput and/or m\_getfilename are called each time an input variable is required, assigning the next element of MEXEC\_A.MARGS\_IN\_LOCAL, or, if it is empty, querying for input. Some functions take lists of indeterminate length, in which case passing the string ‘/’ or ‘ ‘ or ‘-1’ (depending on the function) will terminate the loop to move on to the next type of input.

Many of the functions won’t accept the same file for input and output file, hence the use of temporary wk\*.nc files.

**B. Handy Hints and Tips**

\* If a file crashes in a script it may be left with an "open to write" flag and subsequent scripts will fail. Reset using "mreset".

\* 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.

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.

\* 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.

**C. List of cruise-specific options**

These are the cruise-specific options currently included in the Mexec scripts. Not all of these need to be set for every cruise; many have default values, assigned in **get\_cropt.m**. Look in the **opt\_*cruise*.m** files for examples of settings for each of these parameters, and in the scripts corresponding to each case for how they are used. Recall that oopt is an optional string variable (used to distinguish between multiple calls to **get\_cropt.m** in a given script)

|  |  |  |  |
| --- | --- | --- | --- |
| *CTD data processing* | | | |
| *scriptname* | | *oopt* | *what it does* |
| mctd\_02 | | corraw | edits to be applied to raw file |
| mctd\_02b | | hyst | oxygen hysteresis parameters and function call |
| mctd\_03 | | 24hz | edit 24hz for instance to replace fouled scans from one CTD with data from another |
| 1hz |  |
| psal | exclude and interpolate over bad scans for a particular set of parameters |
| s\_choice | set primary salinity sensor and list of stations on which to use alternate as primary |
| o\_choice | same for oxygen |
| mctd\_04 | | pretreat | remove some data before averaging to 2db; or use upcast data for 2db (on some station(s)) |
| mdcs\_03 | | vstring | single oxygen sensor or two oxygen sensors |
| mfir\_03 | | fillstr |  |
| mwin\_01 | |  | set acceptable time window by station |
| mwin\_03 | |  | fix some winch wireout data when underway logging missing |
| mctd\_checkplots | | pf1 | list of parameters to plot |
| sdata1 | salinity to plot (psal vs asal) |
| odata1 | oxygen to plot (1 or 2 sensors) |
| mctd\_rawshow | | pshow5 | parameters to plot on one figure |
| pshow2 | parameters to plot on another figure |
| pshow4 | parameters to plot on a third figure |
| mctd\_rawedit | | badscans | set scans to edit out of raw data (rather than choosing graphically) |
| pshow1 | parameters to plot together for editing |
| populate\_station\_depths | | fnin | input text file list of station depths |
| bestdeps | edit some of these depths |
| smallscript | | klist | list of stations to batch process |
| *Sample data and sensor calibrations* | | | |
| *scriptname* | | *oopt* | *what it does* |
| ctd\_evaluate\_sensors | | tsensind | set station numbers on which different primary and secondary sensors were used |
| csensind |
| osensind |
| cond\_apply\_cal | | switch on sensor to set conductivity calibration factor as a function of station, pressure, and temperature | |
| oxy\_apply\_cal | | switch on sensor to set oxygen calibration coefficients alpha (function of station) and beta (function of pressure) | |
| numoxy | |  | number of oxygen sensors present |
| temp\_apply\_cal | | switch on sensor to set temperature offset | |
| tsgsal\_apply\_cal | |  | set salinity offset |
| fluorcal | |  | set calibration function for fluorescence |
| msal\_standardise\_avg | |  |  |
| mbot\_00 | |  | default Niskin numbers |
| mbot\_01 | | infile | full path to bottle csv file |
| botflags | default Niskin bottle flags |
| mcfc\_02 | | infile1 | input data file |
| cfclist | list of types of cfcs measured |
| msbe35\_01 | | flag | flag bottles which might have closed too quickly for a good sbe35 reading |
| msal\_01, mtsg\_01 | | salcsv | sets input file name |
|  | | cellT | set cellT if not in file |
|  | | offset | set offset if standards or offset are not in file |
|  | | flag | set bottle/bottle reading flags by station and (Niskin) position |
|  | | indata |  |
|  | | sstdagain | run msal\_standardise\_avg a second time? |
| msal\_standardise\_avg | | std2use | set standards readings to exclude |
| sam2use | set sample readings to exclude; set sample bottle quality flags |
| mtsg\_medav\_clean\_cal | | smdiff | load smoothed differences saved by mtsg\_bottle\_cleanup, to use for calibration |
| mtsg\_bottle\_compare | | dbbad | exclude bad sample data (or bad comparison) |
| sdiff | smoothed differences |
| mtsg\_cleanup | | kbadlims | sets of start and end times of bad data to NaN |
| vout | change from default (which is to just NaN all variables between kbadlims); this is also the place to do something like NaN a given variable when it is out of range |
| moxy\_01 | | oxycsv | set input file name |
| oxybotnisk | translate from bottle rows in the oxygen spreadsheets to Niskin numbers |
| flags | set flags by station and (Niskin) position |
| moxy\_ccalc | | oxypars | set parameters for computing oxygen concentration from titre |
| blstd | blank and standard titre volumes |
| botvols | sample bottle volumes file |
| *Summaries* | | | |
| *scriptname* | *oopt* | | *what it does* |
| mcchdo\_01 | expo | | WOCE expo code and section ID for hydro section |
| outfile | | file to write exchange-format bottle sample data |
| headstr | | header information to write to file |
| mcchdo\_02 | expo | | as above |
| outfile | | file to write exchange-format CTD data |
| station\_summary | optsams | | cell arrays of sample types collected |
|  | stnmiss | | stations not to include |
|  | cordep | | corrected depth field |
|  | comments | |  |
|  | altdep | |  |
|  | varnames | | standard variable names (the optional samples will be appended) |
| *Underway data* | | | |
| *scriptname* | *oopt* | | *what it does* |
| mday\_01\_clean\_av | cnav\_fix | | by default the function cnav\_fix will be applied to the cnav stream to correct an error in labeling minutes as decimal degrees; if this is not necessary (i.e. already fixed), the correction can be switched off |
| morecorr | | use the Mexec short name (called abbrev in this script) to switch on non-standard cleaning/calibration operations |
| msim\_plot | sbathy | | file of atlas bathymetry |
| mem120\_plot | sbathy | | file of atlas bathymetry |
| vmadcp\_proc | aa0\_75 | | approximate/nominal alignment angle and amplitude for 75 kHz and 150 kHz |
| aa0\_150 | |
| aa75 | | additional (refined) rotation and amplitude corrections  based on btm/watertrk |
| aa150 | |

**D. Known bugs and future changes**

D.1. Bugs

~~If a function is interrupted, some fields of MEXEC\_A are not cleared properly such that filenames being written to the history file can accumulate.~~ Resolved, jc159.

Possibly relatedly, errors involving MEXEC\_A.MARGS\_OT may come up (clear all, run m\_setup, and try again).

~~Warning about not finding an exact case match for “redef” or something like that (ignore, unless it turns into an error, in which case, restart Matlab).~~ Resolved, jc159, however see next item.

Issues with different versions of matlab/versions of snctools and mexcdf not properly setting row/col dimensions in .nc files. Workaround for v2011a forces underway data to be rows/columns\*\*\* . . . does not work on v2014b. \*\*\*

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

D.2 Planned future changes

Add loopedit?!?

At least: further reducing querying for input when MEXEC\_A.MARGS\_IN has been supplied; hopefully: more normal functional input argument handling (rather than using global variables for all input arguments or prompting for input)

Separate out initial setup of directories, version files etc. from adding paths from setting global variables (?)

More documentation of functions—but **please point out where specifically this is missing!**

Use matlab’s built-in netcdf support; perform more operations in scripts (rather than passing to mexec source functions) and just have functions to read and write mstar format (no need for mapend, mcalc, etc., and probably no need for m\_write\_header, m\_print\_header etc. as separate functions as opposed to optional operations of the basic reading and writing functions)

Possible condensation of mfir, mwin, mdcs scripts (fewer intermediate Mstar files). Generally reducing file i/o by not writing every intermediate step to a file (this is part of the change to built-in netcdf support above anyway).

Updating mtsg\_lagsal, tsglag, mtow\_04, msam\_nutkg, mcfc\_03 to use gsw rather than sw

checking fluor scripts

Script to append despiked bathymetry data (or option in m\_daily\_processing to stop for despiking along the way?)

D.2.a unnecessary functions/scripts in mexec source (i.e. could be done significantly quicker and/or more concisely)

mcrange

pretty much all the special netcdf handling functions

I already got rid of the function whose entire contents was “return” so that’s something

mtruew\_01 seems to undo/redo some of its own calculations? maybe? hard to tell. also it claims to be adding documentation but the docstrings are still unclear.

D.2.b functions/scripts in mexec source that need better documentation

all of them