A User Guide to Mexec

A Matlab-based Bespoke Processing Suite for Ship-based Oceanographic Data

Version 3.1

updated 8-Mar-18

N. Penny Holliday, Yvonne L. Firing

Ocean Circulation and Processes Subgroup

Marine Physics and Ocean Climate

National Oceanography Centre

Southampton

CONTENTS

add time origin to units of time

add ladcp processing to logsheet

copy mexec scripts to ftp site, email steph

add ladcp programs to mexec000

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 *x, 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 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 here, 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. Documentation of the Mexec functions is a work in progress.

This guide was first written by Penny Holliday, and has been updated for v3 by Yvonne Firing.

1.2 What is Mexec?

The Mexec set of Matlab and shell scripts was developed over a number of years by scientists at the UK National Oceanography Centre, Southampton, principally Brian King, with contributions from many others including ...

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 Cruise-specific options:

Program version control in Mexec v3 operates by way of a set of scripts containing cruise-specific options (one such script per cruise). Parameters and variables that might need to change from cruise to cruise (such as which CTD sensor is primary; the calibration functions for conductivity and other parameters; and more) are set by calls to **get_cropt.m**, which
1) sets any defaults using switch/case on the name of the calling script and in some cases string variable oopt;
2) calls **opt_*cruise*.m** to set cruise-specific options, using switch/case on the name of the calling script and (in some cases) oopt; and

3) in some cases, warns about unset options.

1.2.3 Mexec conventions and Mstar file format

Mexec uses global variables for passing arguments to functions.

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. Many of these are described below. You will need to edit m_setup and run it each time you start Matlab. mload and mgetdirs may also be useful. what else.

1.2.4 Directory structure and processing locations

More details on directory structure are given in 2. Note that links made by conf_script_*cruise*.m 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. (In this case the linkscripts in exec/ would need to be edited, but as their function is to copy raw data from the ship's drive, they are not likely to be rerun ashore.)

1.3 Caveats, changes and bugs

Support for newer versions of Matlab and their netcdf interface is incomplete. The programs work on Matlab R2011 (available on the OCP workstations) and probably/mostly on R2014b.

You shouldn't need to edit sw/mexec_v3/source/, except by commenting/uncommenting lines in **mtnames.m** and **msnames.m** (see Section 2).

If you want your edits, including cruise-specific options, to be integrated into Mexec, please send us your scripts at the end of your cruise. To make it easier to keep track of changes, please reserve mexec_processing_scripts and its subdirectories uway, utilities, and summaries for the central Mexec scripts (listed in this guide), and put your other scripts/functions elsewhere.

To submit edits, report bugs or suggest changes, email yvonne.firing@noc.ac.uk.

Please see the lists of known bugs and planned additions in Appendix D first.

2. Setting up a new cruise

i) Remotely mount data filesystems:

On the Cook, mount 192.168.62.12:/home/techsas/Data on /mnt/techsas and 192.168.62.144:/JC145/Specific_Equipment/CTD/data on /mnt/CTD.

On the Discovery, ***

On the JCR, ***

If you are using one of the NOC MPOC OCP seagoing machines, /etc/fstab may already have the mount points listed, allowing them to be mounted by
> mount -a
as root.

ii) Choose the directory level above which the cruise data will be processed, and put the mexec template directory, mexec000, there (or link to it).

The configuration script (iii) will create a subdirectory with your cruise name here, and link to it from cruise. The examples and references below use /local/users/pstar, the standard location on OCP workstations.

iii) Edit and run the configuration shell script to create a skeleton directory structure and copy programs from mexec000:

a) Copy **conf_script_mexec** from mexec000 to **conf_script_*cruise***.

b) At the top of **conf_script_*cruise***, edit the lines setting mexecloc (the directory referred to in ii), cruise, cruiseno, usys, shipudr, shipcdr, and shipldr. Examples are given in **conf_script_mexec**. If you don't have scs or techsas, set usys = ' '.

c) Run **conf_script_*cruise***. This will create a directory structure for your cruise, under $mexecloc; symbolically link to it from $mexecloc/cruise; copy the software to it, and create symbolic links for use by scripts. Directories that depend on the ship underway data streaming system (SCS or TECHSAS) and particular streams available on a given ship/cruise will be made subsequently by **m_setup.m** (see viii).

iv) Edit shell scripts in exec/

These scripts are used to sync data from the remote directories to the processing directory structure, and to create links with Mexec-conventional filenames pointing to these data files. Some customisation is done in the conf_script, but you may still need to edit the linkscripts to reflect file naming conventions in the remote directories.

ctd_linkscript

techsas_linkscript or jcr/scs whatever

vmadcp_linkscript

lad_linkscript

v) Edit template files in templates/

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 /local/users/pstar/cruise/data/mexec_processing/**m_setup.m**.

Most things you should need to modify are near the top of the file, including cruise number, cruise string, and year of the data time origin. Two flags, "quiet" and "ssd", determine how much information will be displayed to the screen.

viii) If necessary, edit:

a) /local/users/pstar/cruise/sw/mexec_v3/source/mtechsas/**mtnames.m** (for TECHSAS) or /local/users/pstar/cruise/sw/mexec_v3/source/mscs/**msnames.m** (for SCS): add lines, or uncomment/comment out duplicate lines, to reflect the stream names available on your cruise. If adding a new type of stream you can decide on the Mexec abbreviation.

b) /local/users/pstar/cruise/data/mexec_processing/**m_setudir.m**: if you added new Mexec stream abbreviations to **mtnames.m**/**msnames.m** (not just new TECHSAS/SCS stream names), 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 /local/users/pstar/cruise/data/mexec_processing/**m_udirs.m** and regenerate it, and the new directories, by running **m_setup.m**.

c) /local/users/pstar/cruise/data/mexec_processing/**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 a cruise-specific options file, /local/users/pstar/cruise/data/mexec_processing/cruise_options/**opt_*cruise*.m**, for your cruise, following **get_cropt.m** as well as the other files in that directory for guidance on format and options. A list of scripts with cruise-specific options is also given in Appendix C.

x) Add /local/users/pstar/cruise/data/mexec_processing (or the equivalent) to your Matlab path in your startup.m (this is found in your home directory/matlab/).

xi) Create a csv file of Niskin bottle firing information, bot_*cruise*_01.csv in /local/users/pstar/cruise/data/ctd/ASCII_FILES/.

It may be easiest to create a file with all bottles for each planned station set to flag 2 (no problems noted), to be edited after each cast when bottles are either not fired (flag 9), or don't trip correctly (flag 4) etc. (refer to the WOCE hydro flags table); or you can add lines to the file as casts are conducted.

3. CTD data and Niskin Bottle Sample data

3.1 Sea Bird data acquisition and processing

3.1.1 Preparation at the start of the cruise

The first step is to select the SBE output variables. It is essential that the output variables include scan and pressure temperature. 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

- Oxygen hysteresis correction: decide whether to use the SBE oxygen hysteresis correction using standard parameters, or whether to derive your own and apply them using Mexec. Look at options in the SBE data conversion program: it is here that the hysteresis correction is applied and you can uncheck that option. Make sure that Mexec script **moxy_02b.m** is edited to match your requirement.

3.1.2 SBE Data Processing

On the CTD logging computer, the SBE Data Processing software was used for initial processing when the cast was finished, by running the following:

> **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. May include hysteresis correction using SBE parameters, but it is recommended to apply this later, in mexec processing. Recommended: name the output files CTD_CRUISE_*nnn*.cnv

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.

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 SBE files (.cnv and .btl files) can be copied to the Mexec processing directory by running **ctd_linkscript**.

3.2 Mexec Data Processing

3.2.1 The file types

ctd_*cruise*_*.nc contain CTD time series or profiles

dcs_*cruise*_*.nc contain information about scans, positions

fir_*cruise*_*.nc contain information about bottle firing times and CTD data

sam_*cruise*_*.nc contain CTD data from bottle firing times along with corresponding sample data

3.2.2 Processing steps that can be done immediately following a cast

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:

msam_01.m creates an empty sam file, sam_*cruise_nnn*.nc, based on sam_*cruise*_varlist.csv, or, if a file for a previous station already exists, **msam_01b.m** copies it;

mctd_01.m reads 24Hz CTD data into ctd_*cruise_nnn*_raw.nc, including the operator-input position from the .cnv file header;

mctd_02a.m renames SeaBird variable names in ctd_*cruise_nnn*_raw.nc based on templates/ctd_*cruise*_renamelist.csv, and updates the header positions based on the underway techsas or scs streams;

mctd_02b.m carries out oxygen hysteresis correction using SBE default parameters or user's preferred parameters (edit as appropriate, check it matches your decision for SBE data processing), and creates ctd_*cruise_nnn*_24hz.nc;

mctd_03.m averages data to 1Hz (output to ctd_*cruise_nnn*_1hz.nc) and calculates derived variables (output to ctd_*cruise_nnn*_psal.nc);

mdcs_01.m creates empty dcs file which will store information about start, bottom and end of good data in CTD file;

mdcs_02.m populates dcs file with data to identify bottom of cast.

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). Start and end scans are written to dcs_*cruise_nnn*.nc.

>> stn = *nnn*; ctd_all_part2

ctd_all_part2.m calls the following:

mctd_04.m extracts downcast data from psal file using index information from dcs file (created by mdcs_01, mdcs_02, and mdcs_03g); 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 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.

>> 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,

>> stn = *nnn*; mctd_rawedit % and follow prompts

mctd_rawedit.m is a graphical interface that allows 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. To edit bad data over a range of points or variables (for instance, caused not by isolated sensor glitches, but by persistent contamination in the intakes, or by faults in the CTD wiring or CTD-deckbox communications), it may be easier to edit the mctd_rawedit case in **opt_*cruise*.m** to set these scans to NaN for all variables.

>> 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**).

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.

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. If reading from an ascii file, the first column must be station number. The depths column might be filled in manually based on CTD logsheets, or the ascii file might be generated from processed LADCP files by i.e.

> grep 'bottom found' ladcp/ix/data/DL_GPS/processed/*.log > station_depth_*cruise*.txt

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.

>> 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, 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.

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 ***

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.

>> 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 (if available; set this in opt_*cruise* smallscript case) in the master sample file by calling the following scripts for a list of stations:

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 evidence 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**.

>> 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 tested (before being applied to the Mstar files) by setting precalt, precalc, or precalo, respectively, to 1 before running **ctd_evaluate_sensors.m**.

>> 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.

3.5 Outputting data in other formats

3.5.1 1hz files for LADCP processing

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, …

3.5.2 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

4. Underway Data

4.1 TECHSAS/SCS

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, **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 is called temp_r or temp_m, while the housing temperature (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

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

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 are likely to evolve towards more standard argument parsing, without the querying mode.

MEXEC_A having been declared a global variable by m_setup, it is available within any function. 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 files as well as possibly the proliferation of files like fir_cruise_nnn_blt, fir_cruise_nnn_time, and so on (see D).

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 be fine. If netcdf-related errors about setting preferences occur, 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 named scripts for their use.

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
	oxybotnis	translate from bottle rows in the oxygen

	k	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
	aa150	

		based on btm/watertrk

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.

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)

Matlab netcdf errors about "a preference with that name or group already exists" (workaround: restart Matlab).

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 nuts, co2, cfcs, fluor scripts

Smallscripts for nuts and cfcs

Smallscripts for rerunning calibration sample steps only

Smallscripts for undoing calibrations

Simplify station_depths steps

add to minit (maybe): clear history using m_proghd or similar

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