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. Variables or parts of filenames that must be substituted with values are in italic (e.g. a reference to m_daily_proc(*day*) indicates that the day number should be substituted for *day*; opt_*cruise*.m refers to opt_dy113.m, opt_jc238.m, etc.; ctd_*cruise*_*nnn*_2db.nc refers to one of a set of files, with *cruise* replaced by the cruise designation and *nnn* replaced by the 3-digit form of the station number).

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

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

1.2 What is Mexec?

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

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

Commented [FYL1]: Say something about pstar?

Commented [FYL2]: Source/credit?

Commented [FYL3]: Source/credit?

Commented [FYL4]: Check source/credit

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

JR15003 (2015/16): start of mexec_processing_scripts_v3, with introduction of cruise options files (see 1.2.2).

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

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

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

JC238 (2022): version 4: continued simplification of processing steps, merge of mexec and mexec_processing_scripts along with other related Matlab code into a single git repository, git.noc.ac.uk/OCP/ocp_hydro_matlab, and shifting of (maintained) mexec_exec repository from git.noc.ac.uk/MEXEC/ to git.noc.ac.uk/OCP/.

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

Commented [FYL5]: And not at every stage?

Commented [FYL6]: (the intent is to bring options files from JR18002 up to date with the code at the end of JC238

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 and parameters

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

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

The defaults are split, purely for convenience/readability, across four scripts: 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 setdef*.m scripts contain explanations for each set of options and therefore may be useful to examine, but general users should not edit them. Instead, make changes to opt_*cruise*.m for your cruise – it is recommended that you not duplicate (unchanged) default settings in this file -- and raise an issue (see 1.3) if you think the defaults themselves should change.

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

To determine processing choices that went into a given data file, in addition to the comment field, the metadata includes the commit active when the data file was most recently modified; the setdef_cropt_*.m and opt_*cruise*.m files in that commit thus contain a record of corresponding processing parameters. If you add parameters (e.g. a list of salinity sample flags to modify) not explicitly in opt_*cruise*.m but by having opt_*cruise*.m read in another file (e.g. salflags.txt), you should place that file in your data directory (e.g. in ctd/BOTTLE_SAL) so it will be backed up along with the original input "data" files (e.g. sal_jc238_crate1.csv, etc.).

1.2.3 Mexec conventions

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

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

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

1.2.4 Mstar file format

Mexec generates NetCDF files, with a particular format (set of attributes), referred to here as Mstar files. While these NetCDF files can be read with ncdisp.m and ncread.m (or equivalent programs in other languages), there are specific Mexec scripts for interfacing with them. There are two notable idiosyncracies that may trip up when interacting with Mstar files: a) all data must be numeric; b) their specification of time base and units is non-standard.

A quick guide to interacting with Mstar files is given in Appendix ***.

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. (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 and using Mexec

2.1 Before the cruise

1) Get the two git repositories.

If you have git and a login to git.noc.ac.uk with an ssh key linked to the OCP group, clone the master branch:

```
> cd ${PROGDIR} #wherever you choose to keep the software
> git clone git@git.noc.ac.uk/OCP/ocp_hydro_matlab.git
> git clone git@git.noc.ac.uk/OCP/mexec_exec.git
```

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

> cd ${PROGDIR}/ocp_hydro_matlab
> git branch *other_branch*
> git fetch origin *other_branch*
> git checkout origin *other_branch*
> git pull origin *other_branch*

Otherwise, download the desired branch (probably the master branch) from http://git.noc.ac.uk/OCP/ocp_hydro_matlab and http://git.noc.ac.uk/OCP/mexec_exec, and unpack in ${PROGDIR}.

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

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

> cd ${PROGDIR}/ocp_hydro_matlab

> git branch *cruise*

> git checkout *cruise*

> cd ${PROGDIR}/mexec_exec

> git branch *cruise*

> git checkout *cruise*

3) Add mexec_exec and its subdirectories to your shell path (in .bashrc or equivalent). Add ocp_hydro_matlab to your Matlab startup path.

4) Edit the cruise name and processing base directory (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:

```
> conf_script_mexec
```

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

5) Edit ocp_hydro_matlab/mexec_processing_scripts/m_setup.m.

Most things you (may) need to modify are near the top of the file, including cruise designation (stored as MEXEC_G.MSCRIPT_CRUISE_STRING), and year of the data time origin. The "quiet" flag determines how much information will be displayed to the screen while programs are running.

Commented [FYL7]: add: LADCP, what else?

6) Generate an empty cruise-specific options file, ocp_hydro_matlab/mexec_processing_scripts/cruise_options/opt_*cruise*.m, replacing *cruise* with your cruise reference (in the same form as you set in m_setup.m, e.g. dy113). See below, as well as other files in that directory, for examples.

2.2 On the ship

7) 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. ***

8) Edit shell scripts in mexec_exec/: ctd_syncscript, lad_syncscript (for LADCP), and uhdas_01_linkmerge (if reading UHDAS/CODAS data).

These scripts will be used to sync data from the remote directories to the processing directory structure, and will need to be edited to reflect file naming conventions in the remote directories (e.g. ~/mounts/CTD/Data/CTD_DY113_001.cnv vs. ~/mounts/CTD/Data/Processed/ctd_dy113_001_01.cnv, etc.).

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

9) Set up or link underway data processing

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.

10) 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.

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.

2.3 Using Mexec

Each time Matlab is started, 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 or m_global to regenerate them.

3. CTD data and water bottle sample data

3.1 Sea Bird data acquisition and processing

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

Here is an example from JC086.

```
# name 0 = timeS: Time, Elapsed [seconds]
# name 1 = depSM: Depth [salt water, m]
# name 2 = prDM: Pressure, Digiquartz [db]
# name 3 = t090C: Temperature [ITS-90, deg C]
# name 4 = t190C: Temperature, 2 [ITS-90, deg C]
# name 5 = c0mS/cm: Conductivity [mS/cm]
# name 6 = c1mS/cm: Conductivity, 2 [mS/cm]
# name 7 = sal00: Salinity, Practical [PSU]
# name 8 = sal11: Salinity, Practical, 2 [PSU]
# name 9 = sbeox0V: Oxygen raw, SBE 43 [V]
# name 10 = sbeox0Mm/Kg: Oxygen, SBE 43 [umol/Kg]
# name 11 = sbeox0ML/L: Oxygen, SBE 43 [ml/l]
# name 12 = xmiss: Beam Transmission, Chelsea/Seatech/WET Labs CStar [%]
# name 13 = flC: Fluorescence, Chelsea Aqua 3 Chl Con [ug/l]
# name 14 = turbWETbb0: Turbidity, WET Labs ECO BB [m^-1/sr]
# name 15 = altM: Altimeter [m]
# name 16 = scan: Scan Count
# name 17 = ptempC: Pressure Temperature [deg C]
# name 18 = pumps: Pump Status
# name 19 = latitude: Latitude [deg]
# name 20 = longitude: Longitude [deg]
# name 21 = flag:  0.000e+00
```

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

3.1.1 SBE Data Processing

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

2) 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.
3) Cell Thermal Mass to correct the pressure and conductivity. Recommended: set the output name to _ctm so that, for input CTD__CRUISE_nnn_align.cnv, this step will produce CTD_CRUISE_nnn_align_ctm.cnv.

The first and last .cnv files (that is, original and _align_ctm), as well as the .bl and .ros files, should be copied to /local/users/pstar/cruise/data/ctd/ASCII_FILES, while .hex, .hdr, and .XMLCON files should be copied to RAW_CTD_FILES. On unix/linux systems you can use (mexec_exec) ctd_syncscript to do this after editing cruise name and location of original files.

3.2 Mexec CTD data processing

3.2.1 Output file types

ctd_*cruise_nnn_**.nc contain CTD time series or profiles, with different stages of processing, editing, and averaging (see below).

dcs_*cruise_nnn*.nc contains information about scans (start, bottom, end of cast) and positions

fir_*cruise_nnn*.nc contains information about bottle firing times and corresponding CTD and winch data

sam_*cruise*_all.nc is a combined (all-station) file containing the data from the fir_*cruise_nnn*.nc files along with analysed bottle sample data.

win_*cruise*_nnn.nc contains winch information***

3.2.2 Processing steps to do immediately following a cast

The basic steps for CTD processing following a cast are:
> ctd_syncscript
Copies .hex, .cnv, _align_ctm.cnv, and .bl files from acquisition computer to processing workstation.
>> stn = *n*; ctd_all_part1 % *n* is the integer station number
ctd_all_part1.m calls the following:
mctd_01.m loads and renames variables (as set in varlists*** and cruise options files) and saves in ctd_*cruise_nnn*_raw.nc.

mctd_02.m does conversions, edits, and corrections as set in cruise options files, calling ctd_apply_autoedits.m, ctd_apply_oxyhyst.m, select_calibrations.m, and apply_calibrations.m to produce ctd_*cruise*_*nnn*_raw_cleaned.nc and ctd_*cruise*_*nnn*_24hz.nc files. The only default action is to apply (the manufacturer default) correction for oxygen hysteresis, but examples are available for code to remove out of range values, certain scan ranges, spikes, and/or times when the pumps were off; it is generally not recommended to apply these before first examining data (as problems may be masked). Once sample data are available, the mctd_02 case of the cruise options file is also where calibration functions can be specified (with a setting to apply to all stations or only a subset).

Commented [FYL8]: Where to explain difference between manufacturer cal and "our" sample-comparison cal?

mctd_03.m selects primary sensors (as set in cruise options files), computes derived variables (e.g. salinity) and averages to 1 Hz, using grid_profile.m, producing ctd_*cruise*_*nnn*_psal.nc.

Commented [FYL9]: Add information on default choices for filling gaps

mdcs_01.m guesses start and bottom of cast and saves in dcs_*cruise*_*nnn*.nc.

>> stn = *n*; mdcs_03g

mdcs_03g.m brings up a GUI for selection or confirmation of cast start, bottom, and end based on P, T, C, and pumps flag; any modifications are added to dcs_*cruise*_*nnn*.nc.

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

Commented [FYL10]: Explain option to account for oxy_align time offset?

Edit opt_cruise.m for Niskin flags and depths***

>> stn = *n*; ctd_all_part2

ctd_all_part2.m calls the following:

mctd_04.m separates down and up casts, optionally applies m_loopedit.m to the downcast data, and averages to 2 dbar, producing ctd_*cruise*_*nnn*_2db.nc and ctd_*cruise*_*nnn*_2up.nc.

Commented [FYL11]: Info about default and other options

mfir_01.m gets times and scans of Niskin bottle firing from .bl file, along with Niskin bottle numbers (default: 1:24) and flags (default: 2 for all fired bottles, 9 otherwise) from cruise options files, and puts in fir_*cruise*_*nnn*.nc.

mfir_03.m gets CTD data from these scans from the 1-Hz _psal.nc file and adds to fir_*cruise*_*nnn*.nc.

Commented [FYL12]: More info on default and other options

mwin_01.m gets winch information from the underway stream and saves in win_*cruise*_*nnn*.nc.

mwin_to_fir.m adds the winch information to fir_*cruise*_*nnn*.nc.

mfir_to_sam.m puts the data from fir_*cruise*_*nnn*.nc into appended sam_*cruise*_all.nc.

station_summary.m adds position, start and end time, depth obtained by calling best_station_depths.m, and information from sam_*cruise*_all.nc for this station to station_summary_*cruise*.nc and table station_summary_*cruise*.txt. Rather than storing station depths in a text file, they are calculated by best_station_depths.m. Immediately following a

cast, it will try to calculate them from CTD+altimeter; later you can specify to include information from the LADCP (if available), but in any case, if you have casts that are not full-depth, you should use the underway bathymetry data to fill in a list of bottom depths under the best_station_depths case in opt_*cruise*.m.
mdep_01.m adds the depths to the various ctd_*cruise_nnn_**.nc files.

>> stn = *n*; mctd_checkplots

mctd_checkplots.m produces a set of plots to check for sensor drift or other problems. It will make plots comparing the two sensors, plots comparing up- and downcast data. It will also make plots comparing station *n* with other stations, first querying for either some number of preceding stations (including 0) or a list of specific station numbers to use. If you notice loops/static instabilities here, you can activate the cruise option to apply loop editing in mctd_04.m.

>> stn = *n*; mctd_rawshow

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

1) specify automatic edits (based on scan ranges, data ranges, behaviour if pumps go off, and despiking) in opt_*cruise*.m under mctd_02.m case; and/or
2) >> stn = *n*; mctd_rawedit
   mctd_rawedit.m brings up a GUI for selecting and deleting spikes in the data (starting from ctd_*cruise_nnn*_raw_cleaned.nc if available, ctd_*cruise_nnn*_raw.nc otherwise), and saves the selections to ctd_*cruise_nnn* _raw_cleaned.nc, as well as recording them in (text file) mplxyed_*yyyymmdd_HHMMSS*_ctd_*cruise_nnn*

If there are significant instrument or cable problems, the inspection may show spikes in pressure or temperature that are large enough to affect the other data streams, requiring reprocessing from before the cellTM stage*** details in ***.
If automatic edits were added or mctd_rawedit was used,
>> stn = *n*; ctd_all_postedit

ctd_all_postedit.m reruns mctd_02, mctd_03, mctd_04, mfir_03, and mfir_to_sam to apply edits and propagate changes to the ctd_*cruise_nnn*_raw_cleaned.nc file through to other files.

If necessary, iterate the rawedit and postedit steps, and/or mctd_checkplots.m, ctd_all_part2.m to loopedit.

Note: If you need to restart processing from the mctd_01 stage after doing manual edits in mctd_rawedit, to reapply them run ctd_apply_autoedits.m.

3.3 Water Bottle Sample Data

3.3.1 Loading data

The result of the sample data processing is to create a master sample data file, sam_*cruise*_all.nc, populated with CTD firing data, sensor data, and subsequently the water sample data as they become available. The CTD winch, firing and sensor data are pasted into this file during running of ctd_all_part2.m, as described above. This section describes how the water sample data are included in the process.

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

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

Scripts including msal_01.m, moxy_01.m, mnut_01.m, mco2_01.m, mcfc_01.m, miso_01.m (etc.) perform the following steps, depending on parameters and code specified in opt_*cruise*.m:

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

The key information to determine early in the process is:

a) The format and column headers/variable names of the input sample files
b) whether information from file headers is required (e.g. salinometer bath temperature and standard seawater K15 value) and can be read in or should be coded into opt_*cruise*.m.
c) whether you will want to recalculate variables or whether concentrations and salinities will be provided as-is by analysts. For salinity, it is recommended that you read in conductivity ratio values for both samples and standards, and inspect them (using msal_01.m) before coding and applying standards offsets and then calculating conductivity and salinity.

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

3.3.2 Checking data

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

checkbottles_01 plots a single sample parameter for multiple stations, as well as its anomaly (either from the mean, or, if a gridded section file, ctd/grid_cruise_*.nc, has been produced, from the gridded profile), as a function of pressure and as a function of temperature; a graphical user interface enables selection and printing of outliers to be flagged in opt_*cruise*.m or investigated further.

checkbottles_02 plots multiple parameters for a single station, overlaying sample values on CTD or gridded (see above) profiles and labelling Niskin numbers. This script is particularly useful for detecting leaking or closed-at-the-wrong-depth Niskins, which will exhibit bad values for multiple parameters (including, potentially, obvious outliers in oxygen draw temperature).

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

Commented [FYL13]: multiple sensors during ccruise?

Sample flags set in opt_cruise.m under msal_01 etc. cases, and/or Niskin flags set under the mfir_01 case, can be updated to reflect problems identified here. Flags are then applied to the sam_*cruise*_all.nc file by rerunning the script under whose case they were set (e.g. msal_01, mfir_01).

Flag meanings for Niskin bottles (mfir_01): initial flags should be 2 (bottle good, sampled), 3 (leaking [and did not sample]), 4 (did not trip correctly [wire not released or bottle was seen to snap closed on landing]), 7 (unknown problem [maybe leaking but not so obviously we didn't sample]), 9 (did not sample, no further information). Then after examining data update initial flags of 7 to 2 (good), 3 (several sample values suspicious so probably leaking), or 4 (clearly sample is from a different depth, i.e. did not trip correctly), and add additional 3 or 4 flags as indicated by the data.

3.4. Sensor Calibration

Calibration functions are entered in opt_*cruise*.m under the mctd_02, ctd_cals case. They are entered as strings which describe setting dcal.*sensor* (e.g. dcal.cond1, dcal.oxygen2) as a function of d0.*sensor*, as well as (optionally) d0.statnum, d0.press, d0.temp[1/2] etc. Each string is the value of a field whose name is the cruise name (to prevent applying copy-pasted calibration functions from a previous cruise). Additionally, flags under this case must be set to 1 in order for the calibrations to be applied when mctd_02 is run. More details are given in setdef_cropt_cast.m (under mctd_02, ctd_cals).

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

3.5 Outputting data in other formats

3.5.1 1hz files for LADCP processing

3.5.2 LADCP processing for bottom depth

To run basic processing of LADCP data from cast nnn (after mout_1hzasc has been run):

```
> lad_linkscript_ix # to copy data from network machine
```

```
>> cd ladcp/ix
```

```
>> cfgstr.orient = ‘DL’; process_cast_cfgstr(nnn, cfgstr);
```

This will generate plots as well as matlab files in ladcp/ix/DL_GPS/processed/nnn/

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

```
>> cfgstr.orient = ‘UL’; process_cast_cfgstr(nnn, cfgstr);
```

And both together:

```
>> cfgstr.orient = ‘DLUL’; process_cast_cfgstr(nnn, cfgstr);
```

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

```
>> cfgstr.orient = ‘DL’; cfgstr.constraints = {‘BT’}; process_cast_cfgstr(nnn, cfgstr);
```

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

4.1.1 Data access

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

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

help mrvdas	lists the 'mr' commands (alternates: mtechsas or mscs)
mrlookd	tells you filename, start, end
mrnames	lists mexec 'shortnames', full filenames in cell array.
mrdfinfo winch	provides info about that datastream (eg winch)
mrgaps gyro_s 10s	lists gaps in datastream of more than 10s
mrposinfo([yyyy mm dd hhmm])	gives you position for that time
mrlistit	list segments of data
mrload	load data from specified table/stream and time range

4.1.2 Preparation at the start of the cruise

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 SBE38 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 = [ddd]; uway_daily_proc
```

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

uway_daily_proc.m goes through the list of underway data streams found in mrnames (or mtnames/msnames) finds which ones are present in the rvdas table list or the scs or techsas link directory, and calls mday_01.m to load them, producing a series of daily files from each data stream, located in their individual directories (e.g. bathy/singleb/singleb_jc238_d197_raw.nc). It is also possible to exclude some streams/set a limited list of streams for uway_daily_proc.m, but by default it processes everything found in both mrnames (etc.) and m_udir.m. If no data are available on a day from a given stream, that stream is skipped.

uway_daily_proc.m 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), correcting (especially in TECHSAS) or standardising units strings, applying factory calibration coefficients (i.e. to convert from voltage to physical units) where necessary, searching for and flagging backwards time steps or duplicate times in nav streams, NaNing out-of-range values, correcting echosounder depth for speed of sound variations based on the Carter tables, and averaging bathymetry to 30-s; output files are stream_*cruise*_*dddd*_edt.nc.

Commented [FYL14]: still here? or later?

For bathymetry, when both the singlebeam (EA600/EA640) and multibeam (EM120/EM122) centre beam streams are present, mbathy_av_merge.m is called to paste in the depths from the other instrument for subsequent comparison.

After all days have been loaded and edited, uway_daily_proc.m calls m_uway_append.m to add the days' files to the appended file for each data stream (e.g. sim_cruise_01.nc). The list of daily files appended into the master files is given in the header information of that file.

Finally, uway_daily_proc.m calls mnav_best.m, mwind_true.m, mtsg_medav_clean_cal.m, and (for scs) upate_allmat.m, which calculate new variables from the appended files.

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

4.1.4 Navigation: additional processing

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

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

4.1.5 Meteorology: additional processing

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

Commented [FYL15]: comment check they still are, and are correct!

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 read into both TECHSAS and RVDAS in V rather than W/m^2, so factory calibration coefficients should be entered into opt_*cruise*.m under the mday_01_apply_fcal case.

4.1.6 Ocean surface variables: additional processing

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

By default, temperature and salinity are edited for bad times (e.g. when pumps were off), which can be found using mtsg_findbad.m and added to opt_*cruise*.m under the mtsg_medav_clean_cal case, and averaged to 30 s using mtsg_medav_clean_cal.m. They may then be calibrated by comparison with bottle samples (salinity) or CTD 5-m temperature. TSG salinity samples are read in at the same time as CTD samples using msal_01.m (see Section 3). mtsg_bottle_compare.m helps compare salinity and choose a time-varying calibration which can be added to opt_cruise.m under tsgsal_apply_cal*** case. mtsg_medav_clean_cal.m is then rerun with flag usecal set to 1 to incorporate the calibrated data. Finally, mtsg_surfmet_merge combines the tsg data with the other surfmet variables.

Temperature variables: On TECHSAS, sea surface temperature (that is, temperature at the seawater intake) is called temp_r or temp_m, while the housing temperature (temperature inside the inline CTD, applicable to the conductivity measurements) is called temp_h. On SCS on the JCR, there are two sea surface temperature sensors, sstemp and sstemp2; the conductivity measurement temperature, tstemp, and the fluorometer temperature, sampletemp. On the Cook, temp_housing is the temperature applicable to conductivity, temp_remote that at the seawater intake (actually a little way up the pipe), and sbe35dropkeel_temp the temperature on the hull.

You can re-run mtsg_findbad.m, mtsg_bottle_compare.m, and m_tsg_medav_clean_cal.m as many times as required to get a clean record.

If you want to check a calibration already applied, edit the switch at the beginning of mtsg_bottle_compare.m from 'uncal' to 'cal' and rerun.

4.1.7 Bathymetry: additional processing

Following m_daily_proc.m, bathymetry data can be cleaned by interactive scripts *** renovate/mbathy_plot.m; needs checking/updating *** which allows 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, rerun m_uway_append.m for all days for these two streams.

4.2 VMADCP

4.2.1 Acquiring and processing/editing data

4.2.1.1 Vmdas plus CODAS

Vessel mounted ADCP data processed using the old University of Hawaii CODAS software (Matlab/Python hybrid version, see /local/users/pstar/cruise/sw/uh_adcp/programs/index.html for documentation) can be loaded into daily and appended Mstar files using mcod_01.m, mcod_02.m, and mcod_mapend.m. The latter sorts by time, so sequences can be added in any order or any number of times.

The full round of processing from VMDAS to Mstar, including the calls to CODAS quick_adcp.py and gautoedit.m, can be run for a sequence or set of sequences using wrapper scripts vmadcp_proc.m and vmadcp_edit.m, as follows:

>> doall = 1; vmadcp_proc % runs vmadcp_linkscript to sync and link files; writes q_py.cnt using initial angle and amplitude set in opt_cruise.m, and calls quick_adcp.py to load data into database; calls mcod_01, mcod_02, mcod_mapend

> This script will prompt for the instrument to use (75 or 150, probably), sequence number or vector of sequence numbers, and possibly the mode (narrowband or broadband), if not set in opt_cruise.m
>
> If this fails on "cannot find asetup", it may indicate there's not enough data in the sequence

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

> At some point, examine cal/botmtrk/btcaluv.out and cal/watertrk/adcpcal.out (in whichever sequence directories they are found; they will not be generated for every sequence), to refine the angle and amplitude calibrations. Add these to the vmadcp_proc aa75 and/or aa150 cases in opt_cruise.m. Once you have done this, you can rerun vmadcp_proc.m, setting doall = 2:
>
> >> doall = 2; vmadcp_proc % writes q_pyrot.cnt using angle and amplitude set in opt_cruise.m and calls quick_adcp.py to apply angle and/or amplitude corrections; calls mcod_01, mcod_02, mcod_mapend

>> vmadcp_edit % writes q_pyedit.cnt, calls gautoedit.m to enable interactive data editing, calls quick_adcp.py to apply edits; calls mcod_01.m, mcod_02.m, mcod_mapend.m

4.2.1.2 UHDAS plus CODAS

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

0) If you previously applied calibrations (angle, amplitude, xducer_dx or _dy) to data in postprocessing:
rm -rf
/local/users/pstar/cruise/data/vmadcp/postprocessing/DY113/proc_editing/os*

1) uhdas_01 #syncs data from acquisition computer to koaeula

2) uhdas_02 #syncs to postprocessing/proc_editing (and makes links if necessary)

3) uhdas_03 #copies previously made edits (archived in proc_archive) to proc_editing so they will be applied to newly expanded dataset; adds new data to dataset

4) cd ~/cruise/data/vmadcp/postprocessing/DY113/proc_editing

5) cd os150nb
6) dataviewer.py -e & #edit using selectors and/or thresholds; remember to apply edits to every segment of time series
7) quick_adcp.py --steps2rerun apply_edit:navsteps:calib --auto
8.1) cat cal/watertrk/adcpcal.out
8.2) cat cal/watertrk/guess_xducerxy.out
8.2) cp -Rp . ../os150nb_a
9) quick_adcp.py --steps2rerun apply_edit:rotate:navsteps:calib --rotate_amplitude amp --rotate_angle ang --xducer_dx dx --xducer_dy dy --auto #if adcpcal.out shows amplitude or angle calibration needed
10.1) cd ..
10.2) dataviewer.py -c os150nb_a os150nb & #to see the effect of the edits and calibration. If you are not happy with them, repeat 6)-9). When you are happy:
10.4) rm -rf os150nb_a #only keep the edited, calibrated version here now

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

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

14) uhdas_04 #generates .nc files in ~/cruise/data/vmadcp/DY113/postprocessing/proc_edit/os75nb/contour/ and os150nb/contour/

15) uhdas_05 #syncs proc_edit to proc_archive

4.2.2 Output station data for LADCP processing

In matlab

16 – 19) vmadcp_stations_to_ladcp(start_station, end_station)

16) os = 75; nbb = 1; mvad_01
os = 150; nbb = 1; mvad_01
%makes mstar format files containing vmadcp data in
/local/users/pstar/cruise/data/vmadcp/mproc/

17) stn = nnn; cast = ‘ctd’; os = 75; nbb=1; mvad_03
stn = nnn; cast = ‘ctd’; os = 150; nbb=1; mvad_03
%vmadcp upper ocean velocity profiles corresponding to ctd station nnn; run for each station for which new or edited vmadcp data are available

18) mvad_for_ladcp(‘ctd’,nnn,75,1)
mvad_for_ladcp(‘ctd’,nnn,150,1)
%makes file of station nnn data to be used as constraint in LADCP processing

19) cfgstr.orient = ‘DLUL’; cfgstr.constraints = {‘GPS’;’BT’;’SADCP’};
process_cast_cfgstr(nnn, cfgstr);

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 ocp_hydro_matlab/file_tools/mexec / 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, hence the use of temporary wk*.nc files.

B. Handy Hints and Tips

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_global % to set global variables
```

To load an Mstar file:

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

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

To save variables from the workspace to an Mstar file:

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

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

```
>> help_cropt = 1; scriptname = ''; oopt = ''; get_cropt
```

Displays a list of scriptnames and oopts in any of the setdef_cropt_*.m.

Commented [FYL16]: move this to appendix/tips section below

>> help_cropt = 1; scriptname = scriptname; oopt = ''; get_cropt
Displays a list of all calls to get_cropt in file scriptname.m, if this file exists, and if not, of all files which use scriptname = scriptname to call get_cropt.
>> help_cropt = 1; scriptname = scriptname; oopt = oopt; get_cropt
Displays which of the setdef_cropt_*.m contains this (scriptname, oopt) case (and therefore a help string explaining it), along with a list of opt_*.m files which contain it (and therefore examples of modifying the defaults).

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

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

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

Because of the interactive prompting options, sometimes a prompt will appear even if you have passed input arguments in MEXEC_A.MARGS_IN. If Matlab says "Busy|", you don't need to do anything even if it looks like it is asking for input.

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

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

To save data to struct array d, header info to array h:
>> [d h]=mload('filename','/') % '/' means all vars, or you can list the ones you want.

* >> m_read_header(file) allows you to read the header info only

* >> mhistory: returns to you the text you need to copy into a script to recreate the program steps you just ran.

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