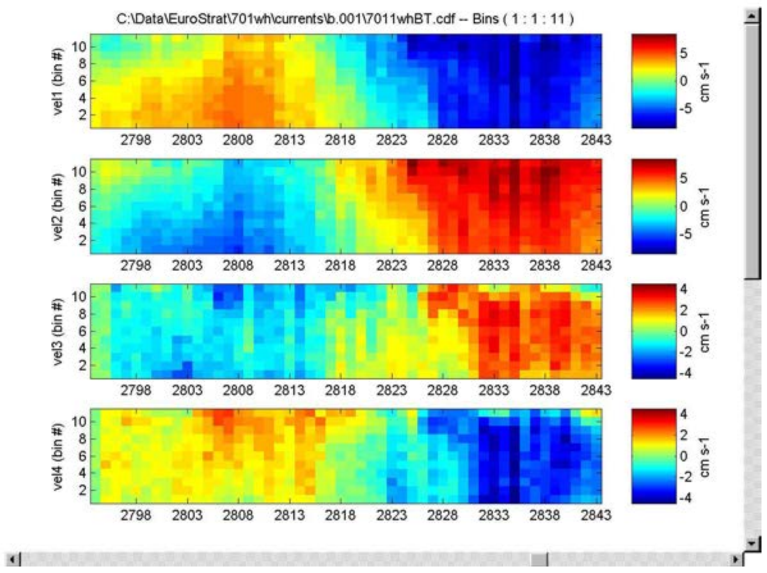
Acoustic Doppler Current Profiler (ADCP) Data Processing System Manual

By Jessica M. Côté, Frances S. Hotchkiss, Marinna Martini, and Charles R. Denham
Revisions by: Andrée L. Ramsey, Stephen Ruane



Open File Report 00–458, version 4

U.S. Department of the Interior
U.S. Geological Survey

U.S. Department of the Interior
KEN SALAZAR, Secretary

U.S. Geological Survey
Marcia K. McNutt, Director

U.S. Geological Survey, Reston, Virginia 2011

For product and ordering information:
World Wide Web: http://www.usgs.gov/pubprod
Telephone: 1-888-ASK-USGS

For more information on the USGS—the Federal source for science about the Earth,
its natural and living resources, natural hazards, and the environment:
World Wide Web: http://www.usgs.gov
Telephone: 1-888-ASK-USGS

Suggested citation:
Côté, J.M., Hotchkiss, F.S., Martini, Marinna, and Denham, C.R., 2011, Acoustic Doppler Current Profiler (ADCP) data processing system manual: U.S. Geological Survey Open File Report 00-458, v. 4, 51 p.

Contents

Figures

Acoustic Doppler Current Profiler (ADCP) Data Processing System Manual

By Jessica M. Côté, Frances S. Hotchkiss, Marinna Martini, and Charles R. Denham

Abstract

This open-file report describes the data processing software currently in use by the U.S. Geological Survey (USGS), Woods Hole Coastal and Marine Science Center (WHCMSC), to process time series of acoustic Doppler current data obtained by Teledyne RD Instruments Workhorse model ADCPs. The Sediment Transport Instrumentation Group (STG) at the WHCMSC has a long-standing commitment to providing scientists high quality oceanographic data published in a timely manner. To meet this commitment, STG has created this software to aid personnel in processing and reviewing data as well as evaluating hardware for signs of instrument malfunction. The output data format for the data is network Common Data Form (netCDF), which meets USGS publication standards. Typically, ADCP data are recorded in beam coordinates. This conforms to the USGS philosophy to post-process rather than internally process data. By preserving the original data quality indicators as well as the initial data set, data can be evaluated and reprocessed for different types of analyses. Beam coordinate data are desirable for internal and surface wave experiments, for example. All the code in this software package is intended to run using the MATLAB program available from The Mathworks, Inc. As such, it is platform independent and can be adapted by the USGS and others for specialized experiments with non-standard requirements. The software is continuously being updated and revised as improvements are required. The most recent revision may be downloaded from: http://woodshole.er.usgs.gov/operations/stg/Pubs/ADCPtools/adcp_index.htm The USGS makes this software available at the user's discretion and responsibility.

Section 1. List of Modifications

The following is a list of major additions and revisions that have been made to the Acoustic Doppler Current Profiler Data Processing System since the third release, which was in September 2005.

Additions:

- Added the ability to use structures and a script file to run the toolbox non-interactively.
- Treats the attribute WATER_DEPTH as a metadata element which gets updated later if surface track and pressure information are available.
- Added the ability to use an m-file to perform surface detect on the ADCP backscatter data. Writes and integrates findsurface.m. The toolbox is now platform independent.
- Added the bindist variable, which contains the distance from the ADCP head to the center of each bin.
- Added range to surface, which is our boundary variable, with data taken from surface detect algorithm used on backscatter data.
- Routinely checks for the problem of ADCP data not indicating the correct orientation when deployed by including fixorientation.m in runadcp.m. Interactive users will be asked which way the instrument was deployed. For script users, the item "settings.deployed_orientation" can now be set.
- Added the coordinate dimension "ensemble" to *.cdf files created by rdi2cdf so that ncBrowse displays the true ensemble numbers in its plots (otherwise it will use a simple index and show the dimension name).
- User has a choice in how close to the surface bins are trimmed as a fraction of the total water depth (1 = water depth, 1.25 = 125% of water depth)
- Created the pressuremask.m function, which will mask points above the surface following (a) pressure sensor data, (b) surface output, or (c) user- provided surface time series. This also serves as an example of how to create code to mask data.
- Created the function maskADCP to replace starbare and allow masking of data via a graphical user interface, which is more compatible with later versions of MATLAB.
- Added two more Demo data sets, one a moored, down-looking ADCP with bottom track data.
- Added the function splitadcp.m to separate current velocity data from wave packets and removed dependence on TRDI tools that work only in the Windows operating system.

Revisions:

- Adapted to changes in wavesmon by TRDI. Wavesmon v3.04 would output currents files (*.PD0) in beam coordinates, but wavesmon v3.06 (and possibly later versions) forces the same data into earth coordinates. This is not a problem for these programs; however, extra caution is advised when applying declination corrections. Rdi2cdf.m no longer corrects the heading; this is done only in adcp2ep.m. In adcp2ep.m, user must supply both a declination value to be stored as metadata and a value by which the vectors and heading shall be rotated.
- Removed automatic start and end ensemble trimming; user will now enter first good and last good ensemble even when interactive.
- Removed starbare, starbeam and starbuck viewers, as they are no longer stable in more recent versions of MATLAB.
- Updated metadata.
- Provided a more detailed discussion in this document about the application of declination.
- Recalculated depth of bins in fix orientation.m so that depths turn out correctly in the final processed file.
- Detects the presence of wave packets data and warns the user to split the data.
- Simplifies handling of magnetic variation corrections to make it easier to apply these to current data generated by waves deployments.
- Clarifies definitions, names, and contents of WATER_DEPTH, instrument_depth, height, range to boundary, and so on, and adds more meaningful explanatory comments.
- Modified trimbins.m to do all the work of finding the ADCP's depth, either by the pressure sensor, range to surface, or user input, and set the bin depths.
- Removed tilttrim.m because it is superseded by findsurface.m in trimbins.m, which uses beam angles, rotations, pitch, and roll to estimate range to surface, and therefore should work at extreme angles.
- Removed presto, proxy, form, and epic code parts, as they do not work well with the most recent version of MATLAB and are not compatible with MATLAB's new embedded NetCDF interface.

Section 2. Overview

This section introduces the USGS fixed-deployment ADCP data processing system and provides information about the history of the system's development. The USGS philosophy regarding data quality is discussed to provide the user with an understating of the motivation behind the system. General information about the following topics will also be discussed: hardware and software required for the system, basic processing steps, limitations, and features that are unique to the programs.

The Sediment Transport Instrumentation Group (STG) at the USGS Woods Hole Coastal and Marine Science Center has a long-standing commitment to providing scientists high quality oceanographic data. To meet this commitment, STG personnel review data as well as hardware for signs of instrument malfunction. STG data sets are accompanied by processing histories which detail data processing procedures that modify the data when editing data. The history also enables data to be reprocessed in light of new insight into instrument function and mooring conditions. The USGS ADCP Data Processing System was developed to meet these data quality commitments for Acoustic Doppler Current Profilers (ADCPs) made by RD Instruments, now Teledyne RD Instruments (TRDI, RD Instruments, 2005).

In 1994 the STG began deploying TRDI ADCPs to sample currents in estuaries and coastal and continental shelf regions. ADCP data sets are large and complex, so existing STG data processing software was insufficient. In addition, the current ADCP processing programs did not meet STG quality control criteria. The USGS ADCP Data Processing System permits engineers and scientists to monitor data quality by:

- processing data with interactive critical review;
- preserving data quality indicators;
- preserving minimally processed and partially processed versions of data sets.

The STG usually deploys ADCPs configured as upward-looking and mounted on bottom tripods, also known as fixed deployments. When the ADCPs are moored on stable platforms, the velocity profiles are recorded in coordinates parallel to the instrument beams (beam coordinates), with minimal internal data processing. STG records ADCP data in beam coordinates in an effort to adhere to the USGS philosophy to post-process rather than internally process data. By preserving the original data quality indicators as well as the initial data set, data can be evaluated and reprocessed if desired. For example, beam coordinate data are desirable for internal and surface wave experiments. TRDI ADCPs internally screen velocity data and yield an output variable that represents the percentage of pings that are good enough to be incorporated into each ensemble average. The screening is performed on data recorded in either beam or geographic coordinates and includes verification of a minimum acoustic ping correlation value, a beam-to-beam comparison of echo intensity values, and a maximum error velocity filter. If the ADCP is set to record in earth coordinates (geographic coordinates), velocity data are converted into north, east, and vertical components, and data-quality variables, such as percent good, represent averages of all four beams. When data are recorded in beam coordinates, the data-quality variables are retained for every ensemble of each beam. To increase the accuracy of current measurements, the

STG normally records ADCP data in beam coordinates, employs additional data screening methods during post-processing, and converts the “cleaned” data into earth coordinates. With each processing step a new data file is created, and the original data file is preserved. This allows the user to step back through the files and identify how data were modified.

Hardware and Software

The ADCP data processing system (ADCP Toolbox) consists of a series of M-files written in the MATLAB language by The Mathworks, Inc. and is supported by most computer platforms. The programs have been specifically tested for Windows XP, Linux, and Macintosh OSX. A copy of the ADCP Toolbox can be downloaded from: http://woodshole.er.usgs.gov/operations/stg/pubs/ADCPtools/index.html. It is assumed that the user has a basic working knowledge of MATLAB.

The current edition of the ADCP Toolbox is Version 7.0. This version has been tested with MATLAB version R2009b. The current release of the ADCP Toolbox is no longer being tested with MATLAB versions prior to R2008a, and we no longer support the ADCP Toolbox 7.0 and earlier releases of MATLAB.

The data processing system was developed to address the issues associated with data recorded in beam coordinates, but data recorded in earth coordinates can also be processed using these routines. The input data file must conform to RD Instruments’ PD0 format outlined in Appendix D of the RDI manual (RDI, 2005). This format is binary and specifies the order in which the data appear. During the “deployment” procedure for the TRDI ADCP, a deployment log file is created containing the ADCP hardware settings. A similar log file is recorded by the ADCP during data collection. The information in these files is required by the system to transform the data from beam to earth coordinates.

After initial input, the data are converted to NetCDF (Network common data format http://www.unidata.ucar.edu/software/netcdf/), and the subsequent processed files are stored in the NetCDF format. As of this revision of this document, a NetCDF interface was added to MATLAB which supports NetCDF version 3.6.2. However, a secondary software package is needed from http://mexcdf.sourceforge.net/ to resolve some syntax differences and for backward compatibility.

NetCDF files may be viewed using the ncBrowse software (Denbo, 2006).

Data Terminology

The types of variables referenced throughout this document are defined below. They are required components of the input data file. The TRDI data file comprises three sections:

- *Header data* – information about the data file type, source, structure, and size.
- *Fixed leader data* – the non-dynamic data including the deployment setup parameters and the system configuration.
- *Variable leader data* - the dynamic data recorded by the clocks and sensors that change with each ping.
- *Ensembles* – measurements recorded to memory. Ensembles may represent single pings or averaged groups of pings. Ensembles may be recorded at regular intervals or in bursts.

For each ensemble, four major data variables are recorded for all four beams:

- *Velocity* – velocity measured parallel to the beam in millimeters per second (mm/s).
- *Correlation* – the signal to noise ratio used as a measure of data quality.
- *Echo intensity* – a measure of the signal strength received by the ADCP.
- *Percent good* – the percentage of pings that passed the data rejection criteria.

Once the data are transformed to earth coordinates, the definition of velocity changes. A measurement of error is included:

- *Velocity* – the east-west and north-south components of velocity, and the average of two estimates of the vertical component.
- *Error velocity* – the difference between the two estimates of vertical velocity.

ADCPs produce a time series of vertical profiles of velocity. Internal processing resolves the acoustic Doppler signal into a set of "bins", each of which represents a layer of water defined by the acoustic travel time from the head of the transducer. These are interpreted as depth levels and may include apparent ghost levels above the sea surface that result from reflection of the beams at the surface. The ADCP is normally configured to internally average bursts of pings and record a time series of these ensemble averages. Each of the resulting ensemble records is given an incremental record number as well as a time stamp. The time is recorded in Julian days.

Basic Processing

The first stage of the processing is to transfer the data from the instrument to the computer. In TRDI ADCPs, data are recorded on PCMCIA flash memory cards. The data can be downloaded using the TRDI recover program or copied from the PCMCIA card directly to the desktop.

If the ADCP was set up to record waves as well as currents, then the mean current ensembles must be separated from the wave packets data by using splitadcp.m (recommended), bbsub.exe, parse.exe, or TRDI's wavesmon software.

It is important to establish a naming convention that will be carried throughout the data processing steps. Using either download method will save the files with the deployment file name given in the initial setup of the instrument. The USGS uses a naming convention that incorporates the mooring number, followed by a number indicating the instrument position on the mooring, and letters indicating the type of instrument. As of the spring of 2005, the USGS STG group indicates a TRDI Workhorse ADCP with the letters “wh” and the TRDI Broad Band ADCP with the letters “bb”. In most cases the deployment name given to the ADCP will be in this convention. In the event that the filename is not in this form, the raw data file name should be changed to meet this convention. In each subsequent major processing step, a new file is created. A letter indicating the type of processing that was performed on the date file is added to the end of the original deployment name. In Section 2 the naming conventions used throughout the processing system are demonstrated by example.

The fully processed data files meet the EPIC NetCDF standard developed by the NOAA Pacific Marine Environmental Laboratory (http://www.pmel.noaa.gov/epic/). These EPIC standards provide a convention allowing researchers from different organizations to share oceanographic data without needing translation. The final data file is designated as the Best Basic Version (BBV), indicating that it is clean of erroneous values, has been converted into earth coordinates, and is compliant with the outlined standards. As of the spring 2005, data from the bins closest to the sea surface are included in the BBV, although many are erroneous. For ADCPs mounted on the bottom, the location of the top good bin changes if the water elevation changes, for example, due to tides. Until more sophisticated methods of cleaning the near-surface data are instituted, scientists will need to exercise judgment in interpreting the top layers.

Setup and Installation

Prior to ADCP Toolbox setup, the NetCDF interface package for MATLAB must be downloaded and installed (see Section 1). The ADCP Toolbox is packaged as a single zip file that contains all of the necessary m-files to run the toolbox and additional routines to complement the toolbox. The contents of the zip file should be extracted to a single directory called "ADCP_Tools". Simply add this directory **and all subdirectories** to your MATLAB path, and the toolbox is ready for use.

Program Use

The system was developed for users of all programming abilities and with the intention of being user friendly. The result is a set of routines that requires a significant amount of interaction. It is recommended that beginners run the software in the interactive mode. Most prompts have a default answer provided, and it is essential that user inputs follow the format of the default answer in detail. Section 5 includes some tips to allow advanced users to increase the efficiency of processing data interactively. For this version, an advanced technique of automated processing has been developed, and this technique requires little or no user interaction. Automated processing may be more convenient for advanced users, users with MATLAB expertise, or for users with problem data sets which might be reprocessed several times.

Correcting for the Wrong Orientation

TRDI ADCPs check only once on wake-up to determine the orientation of the instrument for the entire experiment. The ADCP writes a byte to memory with this initial orientation information, and this byte is used by the ADCP Toolbox to determine how to compute depths. Often the ADCP is still on deck on wake-up, lying on its side in a load cage or shipping crate, so an ADCP may have saved "down" as the orientation, when it will be deployed at the bottom looking up. Likewise, a downward-looking surface-moored ADCP might wake up while set upright on its flat base and start pinging in this position before being deployed looking down.

The best solution is to check the ADCP's perceived orientation before processing and then correct the problem. To determine how the orientation byte in the ADCP has been set, use the file details viewer in TRDI's winadcp program (Utilities menu -> File Details) or, within MATLAB, rdi2cdf.m can be run with only the raw TRDI data file name as an input to read out the TRDI header information.

The toolbox automatically checks the orientation. Interactive users will be prompted to verify the orientation of the ADCP as it was deployed. Script users can set the orientation in the "settings.deployed_orientation" item. The function fixorientation.m will verify the raw NetCDF file's setting with the user's input, change the orientation attribute if necessary, and recalculate the depths accordingly.

Considerations for Waves Deployments

Splitting the Data

TRDI ADCPs that are configured to collect wave and current data simultaneously are able to commingle the measurements from the two sampling schemes and two data formats. It is best to treat these as two separate data streams, wave data and current velocity data, for processing and analysis. These data must be split into wave data and current velocity data before using this toolbox. There are four ways to do split the data, listed below in order of preference.

1. Splitadcp.m is an m-file included with this toolbox which will separate the current velocity ensemble data from the wave packet data. It will preserve the data as they were recorded by the instrument, which may be important for users looking for beam coordinate data. This is the recommended method to use, and is recommended even if wavesmon is being used to process wave array data.
2. Parse is a short program that used to be distributed by TRDI with waves software and may no longer be supported. It will split the wave packets and current velocity ensemble data with no processing performed on the ensemble data or problems with short records being generated. Parse generates one file containing all the current velocity ensembles and one file containing all the wave packets data. Parse preserves the coordinates in which the velocities were originally recorded, such as beam coordinate data. Parse does not shut down cleanly, however. When the split is finished, it must be shut down by the Windows Task Manager. Contact TRDI Field Service for a copy of parse.
3. The wavesmon program, which is used to process waves data, may be used to split the data. This program can also export the binary ensemble data in PD0 format for

use in this toolbox. There are differences in how wavesmon 2x and wavesmon 3x versions apply magnetic variation corrections to the data. Therefore, it is recommended, for simplicity, that the user run wavesmon separately without entering a magnetic variation correction and with all the default settings, just to get the data split. Wavesmon will automatically output PD0 files that will then need to be concatenated (see Getting Started in Section 3 on how to use the DOS copy command to concatenate PD0 files).

4. The bbsub program that can be downloaded as part of the TRDI tools package at http://www.rdinstruments.com will split the data. TRDI field service recommends the setting "do not filter the data." Bbsub preserves the coordinates in which the velocities were originally recorded, such as beam coordinate data. It has been our experience that bbsub can produce data with short records and other problems; however, bbsub is intended to be a replacement for parse and is supported by TRDI.

Correcting for Local Declination

Special care must be taken with the application of magnetic declination. Declination can be specified in any or all of three points during the data collection and processing procedures, so if the user is not careful, data can be corrected several times for the same declination. Figure **1** shows the points at which the user can apply extra rotation to the heading and velocity vectors.

- *During data collection:* The instrument will apply the correction set by the EB command (heading bias) to the heading. Earth coordinate current vectors (only) will therefore be rotated by the value set by EB. If EB was set to the local declination value, then the resulting heading recorded by the ADCP will be the true heading. Ship and instrument coordinate current vectors will not be rotated. Beam coordinate data have not been translated to current vectors at this point.
- *When running WAVESMON* (this is not recommended): If running wavesmon to split the wave and current data, run wavesmon without a magnetic variation correction to get the PD0 files for this toolbox, then run it again with a magnetic variation for wave output with a correction applied.
- *During ADCP Toolbox processing:* The ADCP Toolbox can apply a correction to the heading and velocity vectors in addition to any corrections applied by the EB command.
    - *Step 1, translation from binary to NetCDF (*.cdf output) by rdi2cdf.m:* rdi2cdf.m takes note of only the EB command and includes this value in the metadata. No corrections to heading or rotations are applied by rdi2cdf.m, a change from Toolbox Version 6.
    - *Step 2, coordinate transformation and rotation (*.nc output) by adcp2ep.m:* This step uses two items input by the user: magnetic_variation_applied, a value which will be applied to the heading and vector rotation, and magnetic_variation_at_site, a value which is saved as metadata in the file and is not applied to the heading or vectors. Use magnetic_variation_at_site to record the declination at the location where the ADCP was deployed.
        - Data recorded in BEAM coordinates: The magnetic_variation_applied value is added to the heading in the Hdg variable. This Hdg value is

used in the transformation from beam to earth.

- Data recorded in EARTH, SHIP, or INST coordinates: The magnetic_variation_applied value is added to the heading, and the velocity vectors are also rotated.

Magnetic variation is expressed in compass degrees. Positive numbers are used when magnetic north is clockwise from true north (east), and negative degrees are used when magnetic north is counterclockwise from true north (west).

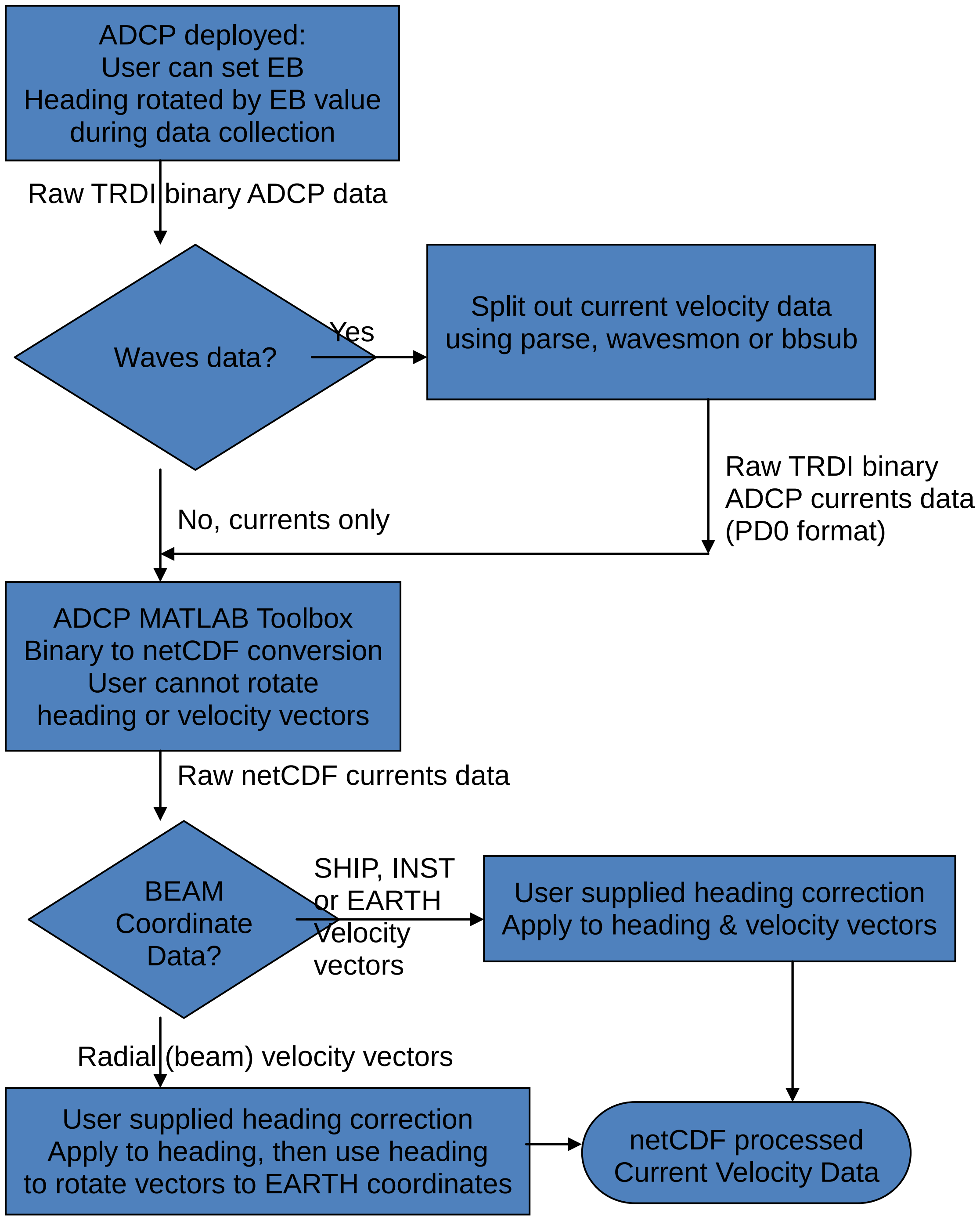

ADCP deployed:
User can set EB
Heading rotated by EB value
during data collection
Raw TRDI binary ADCP data
Waves data?
Yes
Split out current velocity data
using parse, wavesmon or bbsub
Raw TRDI binary
ADCP currents data
(PD0 format)
No, currents only
ADCP MATLAB Toolbox
Binary to netCDF conversion
User cannot rotate
heading or velocity vectors
Raw netCDF currents data
BEAM
Coordinate
Data?
SHIP, INST
or EARTH
velocity
vectors
User supplied heading correction
Apply to heading & velocity vectors
Radial (beam) velocity vectors
User supplied heading correction
Apply to heading, then use heading
to rotate vectors to EARTH coordinates
netCDF processed
Current Velocity Data

Figure 1. Processing flow chart indicating where heading connections are applied.

Special Features

Deployment Log File
When a TRDI ADCP is deployed using the DOS DEPLOY program, a log file is created that contains the codes sent to the ADCP, hardware settings, and the results of each command. The deployment log filename is of the form *.dlg. A log file is also recorded by the ADCP during data collection. This is the "recovery" log file and is of the form *.log. These files contain beam configuration information that is required by the processing routines that transform raw beam coordinate data into earth coordinates. If you cannot locate the *.dlg file, one can be created by using the PS3 command in BBTalk, winSC, or any other terminal emulator program and logging the output (BBTalk uses F3 to log output). The essential component in these files that is required for coordinate transformation has this format:

```
>14 MAR 1999 19:33:07.23 Sent command (PS3) with ADCP response:
Beam Width:   3.7 degrees

Beam    Elevation    Azimuth
  1       -70.00      270.00
  2       -70.00       90.00
  3       -70.00        0.01
  4       -70.00      180.00
```

This feature can be overridden by setting the dialog file name to 'default'. In this case all elevations are set to -70 and the azimuths are 270, 90, 0, and 180 degrees.

If the data are collected in earth coordinates, the coordinate transformation is not needed, in which case a file containing the beam configuration is not requested. Processing data from TRDI Broadband model ADCPs is another special case that does not require a deployment log file and will be addressed more specifically in Section 4.

Bottom Track Data
Bottom track data are extracted by the rdi2cdf.m program in the first processing step that translates the TRDI binary PD0 format to NetCDF. These data are not altered or converted but are available for further analysis. Bottom track settings are stored as global attributes starting with "BT_". Detailed information about bottom track data can be found in TRDI's Workhorse Commands and Output Data Format document.

Section 3. Basic Processing

The basic ADCP data processing system consists of a series of programs that convert data into NetCDF format, add important metadata, edit the data based on quality, time constraints, and depth, transform the data into geographic coordinates, and create an EPIC-compatible data file. The final file created is designated as the Best Basic Version (BBV) of the data.

Although this system addresses several common data problems and works well for most data sets, there will always be exceptions. Data sets that have unusual problems or require further adjustment must be handled on a case-by-case basis and are beyond the scope of this manual.

<u>Overview of Toolbox Programs</u>

I. runadcp.m ~Runs the programs that perform quality checks on the data

Binary Data to NetCDF

A. rdi2cdf.m ~Transfers data from the TRDI output file to NetCDF format
  1. rdhead.m ~ Reads TRDI header data format
  2. rdflead.m ~ Reads TRDI fixed leader data
  3. rdvlead.m ~ Reads TRDI variable leader data

Editing and Quality Assessment

B. fappend.m ~ Concatenates multiple binary files into one file for processing

C. fixEns.m ~ Checks data for missing ensembles and fills in placeholders for any missing ensemble numbers and data

D. trimbins.m ~Removes data bins that represent levels above the sea surface, finds range to boundary and sets bin depths, and adjusts for up-looking and down-looking, and uses one of the following methods to estimate the mean sea level. In order of preference:
  1. pressurecalcs.m ~Uses pressure data to estimate mean sea level
  2. findsurface.m ~Uses a parabolic fit to find the range to surface and thus estimate mean sea level, a good alternative for non-Windows users when pressure data are not available
  3. Ask the user to enter mean sea level and half tidal range.

E. runmask.m ~ Edits data using a mask file that removes data beyond acceptable limits
  1. mkadcpmask.m ~ Creates a file identical in structure to the raw data file but filled with 0's
  2. fillmsk.m ~Sets the limits of the criteria for masking
     a. premask.m ~ Scans the raw data file and marks the mask file for data values that exceed masking criteria
  3. maskADCP.m ~ Allows the user to apply or remove masking interactively via a graphical interface
  4. postmask.m ~ Applies the mask file to the data file, creating a new file

F. pressuremask.m ~Edits data using a mask file that identifies data corresponding to locations above the sea surface, based on pressure, sea-surface height, or user input.
  1. mkadcpmask.m ~Creates a file identical in structure to the raw data file but filled with 0's
  2. fillmsk.m ~Sets the limits of the criteria for masking
     a. premask.m ~Scans the raw data file and marks the mask file for data values that exceed masking criteria
  3. postmask.m ~Applies the mask file to delete identified data and creates a new file

II. adcp2ep.m ~Translates data into variables that are in earth coordinates and creates an

EPIC-compatible data file

Coordinate Transformation
A. runbm2g.m ~Creates the input and output variables to prepare for coordinate transformation
   1. bm2geo.m ~Converts beam data to geographic coordinates

Naming Conventions

We will use examples to illustrate how to run these programs and will employ naming conventions that facilitate the finding and interpretation of data files. As discussed previously, the names are composed of mooring number, position number, and instrument-specific letters. As an example, *9991wh* represents a Workhorse ADCP as the uppermost instrument on the mooring identified by the number 999. A mooring number is a sequential number given to every instrument package that is deployed by the STG group, whether it is a surface buoy, an instrumented mooring, or an instrumented tripod. The ADCP on a tripod will almost always be the uppermost instrument and be distinguished by a 1 for position. Continuing with this example for the naming of data files:

9991wh000.000	Binary raw data file recorded by the ADCP
9991wh.cdf	NetCDF raw data file (or appended files) translated by rdi2cdf.m
9991whF.cdf	NetCDF raw data file with missing values filled in by fixEns.m
9991whT.cdf	Trimmed data file output by trimbins.m, range to boundary data added
9991wh.msk	Mask file output by runmask.m/fillmsk.m
9991whM.cdf	New data file after editing, output by runmask.m/postmask.m
9991whTP.cdf	Trimmed data that have been further edited using pressuremask.m (Note: Downward-oriented data will not have a *TP.cdf file.)
9991wh.nc	EPIC NetCDF file output by adcp2ep.m (BBV)

Getting Started

At this point all hardware and software requirements should have been met and the processing programs and toolboxes should be installed. You are almost ready to begin using the toolbox to process data. *However, if wave data were collected by the ADCP, there is one more step before the ADCP toolbox can be run: separate the wave packets data from the current ensembles. This can be accomplished by running splitadcp.m.*

It is suggested that the raw binary data file be viewed in TRDI's winadcp program. With this program, the user can determine how many ensembles are in the file, the value of the EB setting (which adds an offset to the heading to correct for magnetic variation), and the first good and last good ensembles. Use Utilities, File Details menu selections to check the data file for missing or wrong sized records. Wrong sized records will crash this toolbox.

In the beginning, if you are a novice at using this toolbox or processing ADCP data, use this system in its interactive format. This requires the user to call only the two primary programs, runadcp.m and adcp2ep.m. To prepare for processing, the user should gather pertinent data and files that will be requested during processing (see below). Please review the notes on runadcp.m and adcp2ep.m, and their dependents, to learn what information will be needed. When running these programs it is suggested that the user keep a "diary" file (see “help diary” in MATLAB). The ADCP processing programs output many messages to the MATLAB screen, and a log of these messages will be helpful to diagnose problems if they arise.

The following steps can be used to process your data.

1. Place binary data file(s) and log file in a folder that will be used to save subsequently processed data files. If there is more than one binary file, the files can be appended together using the following procedure:
   - At the start menu select Programs, then Accessories, then Command Prompt to open a DOS window.
   - Type (without quotation marks) “cd c:\mydata\directory” to change to the directory where your raw binary ADCP data files are located, where c:\mydata\directory is the path to those data files.
   - Type “dir *.00*” to verify that the files are there.
   - Use copy to concatenate the files. For example, if you have three files called mydata000.000, mydata000.001 and mydata000.002, then in the DOS command window you would type “copy /b mydata000.000+mydata000.001+mydata000.002 mydataall.000”. Your three files will be appended to each other, in that order, and then saved in mydataall.000. The /b switch is necessary; it tells copy to treat the files as binary files. Concatenating files this way in the beginning is more convenient than doing so later, although the toolbox does have provision to process two ADCP binary files. In LINUX or OSX the equivalent command would be "cat mydata000.000 mydata.001 mydata.002 > mydataall.000".
   - Gather information that will be needed for runadcp:
     - Mooring number
     - Platform type
     - Deployment date
     - Recovery date
     - Nominal water depth at the deployment site, in meters
     - How the water depth was measured
     - ADCP serial number
     - Distance between the ADCP transducers and the sea bed
     - Predicted accuracy of the velocity measurements given by PlanADCP prior to deployment
     - Amount of time the ADCP clock was slow
     - Portion of profile to save, as a percent of the water column
     - The actual orientation of the ADCP as it was deployed
     - First good and last good ensemble numbers (use TRDI’s winadcp)

   - Information needed for adcp2ep:
     - Experiment name

- Project name
- Description of data
- Comments about data
- Longitude in decimal degrees
- Latitude in decimal degrees

2. Start MATLAB and change directory to the folder containing the data files.
3. Type “diary *filename*” at the MATLAB prompt.
4. Type “runadcp” and follow the prompts.
5. At the end of runadcp, the ncBrowse (see Section 4. Display and Analysis Programs) may be used to check over the newly processed data file. If the new data file contains bad values, see Section 6. Troubleshooting.
6. Type “adcp2ep” and follow the prompts.

Your data should now be free of erroneous values and in a format that is ready to be analyzed further. For an average data file that contains 3 months worth of data, these steps could take one to several hours to process (depending on the speed of your computer and the number of ensembles in the data file). Please pay attention to the prompts as well as messages that appear in the MATLAB window. The following pages of this section detail this process. For each function, an example is given that lists the prompts as they appear in the MATLAB window. A double asterisk (**) indicates an instruction to the user, and will not appear as a MATLAB prompt.

The following pages provide summaries and descriptions of each of the functions used by the ADCP Toolbox.

Program:	runadcp.m
Purpose:	To provide the interface for input and output of information required to run the programs that perform the quality checks on the data.
Command:	[theResult] = runadcp (‘rawdata’, ‘rawcdf’, ‘theMaskFile’, ‘theNewADCPFile’, ‘trimFile’)
Description:	This program is used as an organizational tool to gather input file names, build output file names, and check the data types for adherence to the restrictions imposed by this data processing system. All of the dependent programs executed within runadcp.m are stand-alone programs that will be discussed individually below. More information on running this program at the command line is given in the section entitled “Some helpful hints about runADCP.m”.

Most of the prompts described below result from the dependent programs that are doing the processing. Program names are identified in the left column. For these prompts, the notes here are brief. More details can be found in the program descriptions.

Example run:

Program	Prompt	Response
fappend	1. How many binary files used?	1
rdi2cdf	2. Select Binary ADCP File:	9991wh000.000
	3. Save NetCDF ADCP File As:	9991wh000.cdf
	4. Input mooring data for ADCP:	
	A. Enter the mooring number	9991
	B. Enter the platform type	tripod
	C. Enter the deployment date	13-aug-1999
	D. Enter the recovery date	17-aug-1999
	E. Enter the water depth (m)	20
	F. How was the water depth measured?	Ship fathometer
	5. Input metadata from the mooring log	
	A. Enter the ADCP’s serial number	185
	B. Enter the offset of the ADCP transducers from the sea bed in meters	1.25
	C. Enter the predicted accuracy given by PLAN in centimeters per second	1.0
	D. Enter the amount of time, in seconds, the ADCP clock was slow	35
	E. Enter the magnetic variation, degrees west = negative	-16
	F. Enter the initial latitude, degrees north	42.4567
	G. Enter the initial latitude, degrees east	-70.4567
	6. Input for ensembles to process	
	First ensemble	1
	Last ensemble (Inf = all)	Inf
	7. What was the actual deployed orientation?	Up

fixEns	8. Save filled ADCP file as:	9991whF.cdf
	**If no missing ensembles were found, this file will not be used	
trimbins	9. Save trimmed file as:	9991whT.cdf
	10. **A figure appears displaying the depths calculated for each ensemble.	
	Check surface program output:	
	A. Minimum depth in meters:	17.35
	B. Maximum depth in meters:	18.62
	11. Defining Water Depth	
	A. Which method to trim the bins?	
	**This prompt will not appear if pressure sensor installed.	
	Choose: Pressure Sensor, USGS findsurface.m or User Input	
	For User Input:	
	B. mean sea level value	30
	C. units	meters
	D. half the tidal range value	1.5
	E. units	meters
	(**skip remaining prompts for User Input)	
	12. For USGS findsurface.m	
	Select the percentage of water column to keep	105
runmask	13. Create a mask file as	9991wh.msk
	14. Save masked ADCP file as:	9991whM.cdf

Notes on prompts:

Prompt 4: It is very important that the deployment and recovery dates be entered in the exact date format as the default. If this information is not available or appropriate, default or obviously bad values should be used.

Prompt 5:

- C. Prior to deployment, the PlanADCP program (was called PLAN) is used to set the command values given to the ADCP, view the results of the choices, and create a command file. One of the expected values given by PlanADCP is the standard deviation (was called the predicted accuracy) of the horizontal velocity measurement in centimeters per second.
- D. When the instrument is recovered, the ADCP clock is checked against the actual time clock, and the difference between the two is the number of seconds the ADCP clock is slow. This number represents the drift of the internal ADCP clock. If there was a mistake setting the time of the instrument to the correct time format, such as the difference between GMT and EST, the time should be corrected after rdi2cdf.m is run and not reflected in the slow_by attribute. If the ADCP clock is fast (very unusual), use a negative sign to indicate that the clock was fast rather than slow.

Prompt 10: This prompt will only appear if the instrument does not have a pressure sensor installed and is using findsurface.m to estimate a range to surface using acoustic backscatter.

Prompt 11: If ‘Pressure Sensor’ is selected and the software does not detect a pressure sensor, the user will be presented with a list of methods from which to choose. The USGS findsurface.m program is recommended.

Some helpful hints about runADCP.m:

- To minimize the user interaction required by this function, give it a full list of filenames when calling the function. Here is an example: [theResult] = runADCP('9991wh000.000','9991wh000.cdf',’9991whF.cdf’,'9991wh000.msk', '9991whM.cdf', '9991whT.cdf')
- If 9991wh000.cdf is not an existing file, then two dialog boxes immediately appear, asking for the mooring log information. If 9991wh000.cdf already exists, then the program will proceed directly to masking. If there is an existing NetCDF file to use, type in the first two inputs in the command and runadcp will skip rdi2cdf, which is a time-intensive function of the program.
- **If the mask file '9991wh.msk' exists, it will not be rewritten.**
- Depending on the file size, this program could take an hour to run. The window will update itself to indicate the time that has elapsed and the number of ensembles that have been processed.
- If you are trying to start over from scratch, delete all the *.cdf and *.msk files from the directory and type "clear" at the ">>" prompt to clear variables from the MATLAB workspace and ensure that you are making a fresh start at processing.

Program:	fappend.m
Purpose:	fappend.m concatenates two or more ADCP binary files.
Command:	fappend('outfile', 'infile_1', infile_2', ...)
Description:	On occasion, the ADCP will record the data into several raw binary files, instead of one. In these cases, the user may want to concatenate the files before processing begins. This function is used within the runadcp.m and trimbins.m functions.

Example Run:

	Prompt	Response
1.	How many binary files used?	1
2.	Select Binary ADCP File?	999wh000.000
OR		
1.	How many binary files used?	2
2.	Select 1st Binary ADCP File:	999wh000.000
3.	Select 2nd Binary ADCP File:	999wh000.001

Program: rdi2cdf.m

Purpose: rdi2cdf.m extracts the raw data from the binary file and converts and catalogs the data into the NetCDF format.

Command: rdi2cdf(‘rawdata’, ‘rawcdf’, minens, maxens)

Description: The TRDI data records have three parts; header data, fixed variable leader data, and variable leader data. Each of these data types is handled independently by one of the dependent programs: rdhead.m, rdflead.m, or rdvlead.m. These three programs were created to read the exact structure of the TRDI binary data file as detailed in the TRDI manual, Appendix D (2005). It is beyond the scope of this document to provide further details of these dependent programs. When rdi2cdf is run independently of runadcp, the filenames must be provided; the minimum call statement for this function is rdi2cdf(rawdata,rawcdf). We suggest using the rawdata file name with a *.cdf extension for the rawcdf filename, as in the example below. rdi2cdf.m can also be used to create a subset of the full data file by including two additional inputs: minens, the minimum ensemble number, and maxens, the maximum ensemble number. An example use of the full call statement would read theResult = runADCP('9991wh000.000', '9991wh000.cdf', 100,1100).

Example Run: theResult = runADCP('9991wh000.000','9991wh000.cdf').

	Prompt	Response
1.	Input mooring data for ADCP:	
	A. Enter the mooring number	999
	B. Enter the platform type	tripod
	C. Enter the deployment date	01-jan-1999
	D. Enter the recovery date	01-jan-2000
	E. Enter the water depth (m)	20
	F. How was the water depth measured?	Ship fathometer
2.	Input metadata from the mooring log:	
	A. Enter the ADCP’s serial number	185
	B. Enter the distance between the ADCP transducers and the sea bed in meters	1.25
	C. Enter the predicted accuracy given by PLAN in centimeters per second	1.0
	D. Enter the amount of time, in seconds, the ADCP clock was slow	35
	E. Enter the magnetic variation at the mooring location in degrees	-16
	F. Enter the initial latitude, degrees north	42.4567
	G. Enter the initial latitude, degrees east	-70.4567
3.	Input for ensembles to process	
	A. First ensemble	1
	B. Last ensemble (Inf = all)	Inf

Notes on prompts:

Prompt 1: It is very important that the deployment and recovery dates be entered in the exact date format as the default. If this mooring information is not available or useful for the user, default or obviously bad values should be used.

Prompt 2:

- C. Prior to deployment, the PlanADCP (formerly called PLAN) program is used to set the command values given to the ADCP, view the results of the choices, and create a command file. One of the expected values given by PLAN is the standard deviation (was called the predicted accuracy) of the horizontal velocity measurement in centimeters per second.
- D. When the instrument is recovered, the ADCP clock is checked against the actual time, and the difference between the two is the number of seconds the ADCP clock is slow. This number represents the drift of the internal ADCP clock. If there was a mistake setting the time of the instrument to the correct time format, such as the difference between GMT and EST, the time should be corrected after rdi2cdf.m is run and not reflected in the slow_by attribute. If the ADCP clock is fast (very unusual), use a negative sign to indicate that the clock was fast rather than slow.

Other pertinent information:
This m-file writes out a variable 'D', which will holds the distance of each bin from the transducer head for the velocity profile. The actual values in this variable will be corrected and converted into depth as the data go through the other processing steps.

Dependents: rdhead.m, rdflead.m, rdvlead.m, history.m

Program:	fixEns.m
Purpose:	fixEns.m checks the ADCP data for missing ensemble numbers. If missing ensemble numbers are detected, fixEns.m inserts the appropriate ensemble numbers, along with their times, and fill values as placeholders for the missing data. FixEns.m may or may not be successful depending on the nature of the problem; it is limited by interp1.m requirements for monotonically increasing ensemble numbers and can fail to correct complex problems. If fixEns.m cannot solve the problem, then one can use winADCP to split the data into subsections and process the data in separate subsections. The batch functionality of the toolbox allows data to be processed without using fixEns.m, by specifying settings.fixens.run to be 0 (see scriptexample.m).
Command:	[missEns] = fixEns('rawcdf', 'theFilledFile')
Description:	On occasion, the ADCP will not record a data ensemble when measurements fail to meet accuracy standards. fixEns.m examines all the records to see if any ensembles are missing. If all ensembles are present, the function stops. Otherwise, the function will determine the number of ensembles that are missing, the specific ensemble numbers that are missing, and the times of the missing ensembles. The function creates an array of fill values for each variable in the data set and inserts new ensemble records constructed of these arrays, the ensemble numbers, and correct times. This function does not write over the rawcdf file but creates a new NetCDF file.

Example Run:

	Prompt	Response
1.	Select ADCP Data File:	9991wh000.cdf
2.	Select ADCP Filled File	9991whF.cdf

Dependents: history.m

Program:	runmask.m
Purpose:	This program uses the data quality variables measured by the ADCP to edit the velocity measurements. The process uses a mask file, which is preserved as a record of the values that were removed.
Command:	[theNewADCPFile, theMaskFile] = runmask ('rawcdf', 'theMaskFile', 'theNewADCPFile', noninteractive) OR runmask
Description:	runmask.m is the second program called by runadcp with the above command statement. It can be run automatically (with all inputs) or interactively (simply type runmask). If the raw NetCDF file name, the mask file name, or the new ADCP file name is not specified in the call statement, the user will be prompted by dialog boxes. Our convention is to use the unedited data file name for the mask file name, with a .msk extension. Runmask.m creates a mask file which is identical in size to the raw NetCDF file, and fills the variables with 0's. The variables in the raw NetCDF file are then scanned and compared with data quality criteria that are expressed as acceptable limits for the velocity, correlation, echo intensity, and percent good (done by fillmsk.m). If these criteria are not met for a given velocity component, the value 1 is written to the corresponding ensemble and bin in the mask file, indicating a data value that is bad. Runmask.m then calls postmask.m, which produces theNewADCPFile by removing the data values in the rawcdf file that correspond to the bad-data flags in the mask file. Our convention is to give theNewADCPFile a name like *M.cdf. When this function is run interactively (noninteractive = 0), the user is given the option to review the mask file and to mark additional bad data values by hand before the data file is edited. If this function was run successfully, it will produce theMaskFile (*.msk) file and the theNewADCPFile (*M.cdf).

Notes:

- The criteria for data quality consist of acceptable limits for each of the four variables: velocity, correlation, echo intensity, and percent good. These limits are extracted from the settings of the ADCP, which are recorded as attributes in the raw NetCDF data file. It is possible for the user to input the limits for masking, but this requires breaking the programs down into several steps. For more information, see the description of fillmsk.m.

Example run:

	Prompt	Response
1.	Select NetCDF ADCP File:	9991wh000.cdf
2.	Save Masked ADCP File As:	9991whM.cdf
3.	**The maskADCP window opens **The user may now manually add or remove mask.	
4.	**Close the maskADCP window to make sure all mask changes are saved to the *.msk file.	

5. **Tell the runmask function to proceed by pressing return.

Notes on prompts:

Prompt 2:	The ‘000’ portion of the file name is typically dropped at this stage to allow room for additional letters to be added to the file name. These letters indicate the completion of subsequent processing steps.
Prompts 3-5:	Will only appear if runmask is used interactively, with noninteractive set to 0 or fewer than 4 arguments are supplied. See maskADCP for details.

Dependents: mkadcpmask.m, fillmsk.m, premask.m, postmask.m, maskADCP.m, history.m, add_minmaxvalues.m

Program: maskADCP

Purpose: maskADCP is an interactive graphical user interface (GUI) that displays raw ADCP data (*.cdf) and allows the user to edit a mask file (*.msk) consisting of flag values for data quality (at present, only two flag values are used, 0 and 1).

Command: maskADCP('file', 'theRawCDF.cdf', 'mask', 'theMask.msk') or maskADCP alone will prompt the user for the file names

Description: maskADCP is primarily for setting additional data quality flags. It will be called by runmask if runmask is in interactive mode. **MaskADCP does not edit the raw ADCP data file (*.cdf), it edits the mask file (*.msk).** The mask must be applied to the data using postmask.m. The mask file is created by runadcp's call to runmask or may be generated by using runmask.m independently. maskADCP is best used interactively with runmask.m. See example below. Based on user input in the graphic window (Figure 2, maskADCP toggles the values of 1 and 0 to indicate masked (bad) or unmasked (good) data, respectively, in the *.msk file. This *.msk file is applied to the raw *.cdf file in subsequent processing steps to generate the masked data file, *M.cdf. All these files may be viewed with ncBrowse. More about masking is explained in the section documenting runmask.

MaskADCP.m features

- Four plot axes are always displayed, data are plotted in sets such as four beams of velocity, acoustic backscatter, correlation, and percent good. Ancillary data variables such as pressure, temperature, and orientation are grouped into logical sets. Where data are not present, for instance in the case of ADCPs without pressure sensors, the axis will display "empty plot".
- A push button to open or close the mask file (and thus turn on and off the masking of the data display by the mask file) is located in the upper right part of the window in Figure 2. When a mask file is open, and masking is active, the button is red with the name of the mask file displayed.
- A push button in the lower right corner of the figure allows the user to toggle between masking and unmasking data. Thus the user may remove the masking automatically applied by runmask.m. The button is red when masking is being applied and green when being removed.
- Data to be masked are selected interactively by clicking and dragging a box to highlight the data in the axes. Data selected in one axis will be applied to all visible in the window for velocity, acoustic backscatter, correlation, and percent good. For vector time series, the mask is applied to the variable selected. However, once data have been selected, the user will be given the option of masking all variables or canceling the operation. Thus, data selected in the plot are not masked without confirmation by the user. When selecting time series data such as temperature, the mouse must be on the plot line.
- The raw data file being viewed is displayed at the top of the window.
- Along the top of the uppermost plot are the time span and ensemble numbers of the data that are visible in the plot windows.

- At the bottom of the window are two sliders. One allows the user to move along the time series and displays the start and end time and ensemble numbers found in the file. The other slider allows the user to control how many ensembles are displayed in the plot window, allowing the user to zoom in and out. The more data that are displayed in the windows, the slower the rendering of the plots will be. Displaying 3 to 7 days of data is optimum.
- A list box in the lower right corner of the window selects which sets of variables are displayed.
- The user may change the limits of the color plots (and color bars) by selecting "Set Color Bar Limits" in the maskADCP menu.
- Closing the maskADCP window closes the mask and raw data files.

Example run using runmask to generate, manually mask data, and apply the mask:

	Prompt	Response
1.	Select ADCP File:	9991wh.cdf
	**a 9991wh.msk file is generated	
2.	Select Masked File:	9991whM.cdf
	**the 9991wh.msk file is updated based on default standards of good and bad ADCP data	
3.	The maskADCP window opens.	
	**The user may now manually add or remove bad data flags from the mask file.	
4.	Close the maskADCP window to make sure that all changes are saved to the *.msk file	
5.	Tell the runmask function to proceed by hitting return	

Example procedure to apply extra masking when using scripts to process ADCP data:

1. Run your processing script based on scriptexample.m but using only the first step that executes runadcp.m.
2. Run maskADCP.
3. Re-run runadcp, inputting the names of the *.cdf and *.msk files that have already been generated so they will not be generated again. The new masking will be applied.

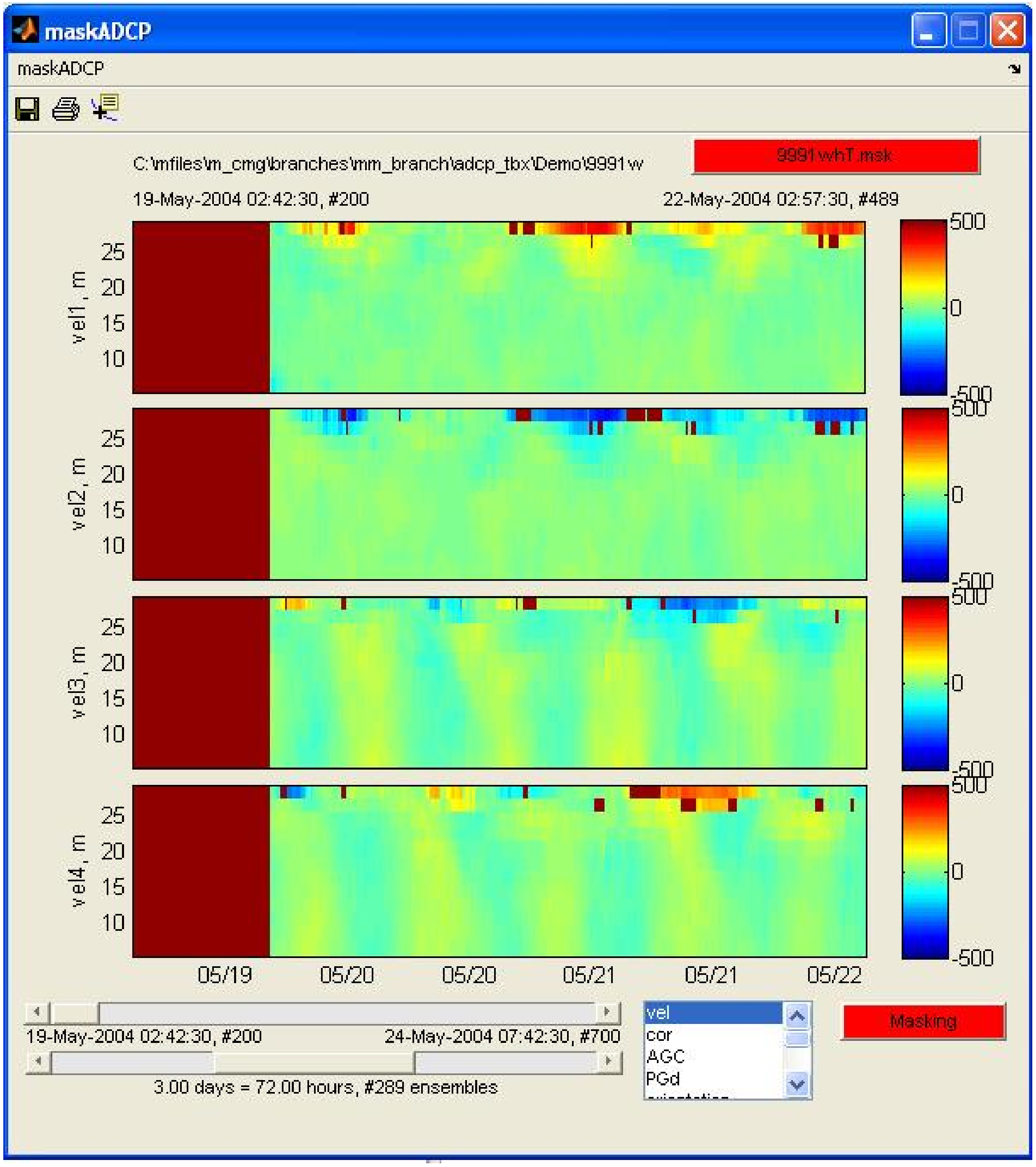

maskADCP
maskADCP
C:\mfiles\m_cmg\branches\mm_branch\adcp_tbx\Demo\9991w
9991whT.msk
19-May-2004 02:42:30, #200
22-May-2004 02:57:30, #489
vel1, m
vel2, m
vel3, m
vel4, m
25
20
15
10
500
0
-500
05/19
05/20
05/20
05/21
05/21
05/22
19-May-2004 02:42:30, #200
24-May-2004 07:42:30, #700
3.00 days = 72.00 hours, #289 ensembles
vel
cor
AGC
PGd
Masking

Figure 2. Screen shot of maskADCP.

Program:	fillmsk.m
Purpose:	Using the minimum and maximum acceptable limits for the four ADCP variables, this program scans the data file and marks the mask file with 1's where the data do not meet the criteria.
Command:	[theMaskFile,velR,corT,echI,Pgd] = fillmsk('theDataFile', 'theMaskFile', velR,corT,echI,Pgd)
Description:	This program is the main engine of runmask.m. The fillmsk call allows a user to specify the acceptable ranges for velocity, correlation, echo intensity, and percent good, but these are optional. Runmask calls fillmsk with minimal input. If no inputs are provided in the call, the program extracts the acceptable minimum and maximum values from the raw NetCDF data file, which records the ranges that are listed in the binary data file recorded by the TRDI ADCP. However, direct use of this program enables the user to input the acceptable limits to be used in scanning the data for bad values. If the user intends to specify variable ranges, all inputs must be defined explicitly, and mkadcpmask.m should be used to create the mask file, prior to running fillmsk. The four variables, velR, corT, echI, and Pgd, are the input variable ranges and require two values each in the form, [min max]. Once the ranges are defined, either by the user or by reading the data file, premask.m is invoked to actually perform the scanning of the data and the filling of the mask file. The following example lists prompts that are seen when no input ranges are given.

Example runs:

	<u>Prompt</u>	<u>Response</u>
1.	Select ADCP Data File:	9991wh000.cdf
2.	Select ADCP Mask File:	9991wh000.msk

	<u>Prompt</u>	<u>Response</u>
1.	Select ADCP Data File:	9991wh000.cdf
2.	Select ADCP Mask File:	cancel
3.	Do you want to create a mask file now?	Yes

Notes on prompts:

Prompt 2:	If a mask file has already been created, at this point it can be selected and submitted. Otherwise, the user should select 'cancel' and the next prompt will appear.
Prompt 3:	This prompt is given when a mask file does not yet exist. If 'yes' is entered, the program will create a mask file with the same name as the input file but with the .msk extension. If 'no' is entered, the program will terminate without creating or filling a mask file.

Dependents: mkadcpmask.m, premask.

Program:	postmask.m
Purpose:	This program applies a mask file to a data file in order to remove data that are flagged as bad in the mask file.
Command:	postmask('theADCPFile', 'theMaskFile', 'theNewADCPFile')
Description:	After the mask has been filled or modified, typically by fillmsk.m or maskADCP.m, postmask.m must be run to modify the beam coordinate data file. (Both steps are included in runmask.m.) The mask file is a logical array where 0's indicate good data points and 1's denote bad data points. When applied to the data file, the mask file acts as a screen and produces a new data file where the bad data points have been replaced with fill values. The fill value used is specified as a variable attribute in the NetCDF file. When postmask is run separately, the command line can contain the three file names, which are self explanatory from the command statement. It is suggested that the 000 portion of the name be dropped for the new ADCP File and an M be added at the end to indicate that the data have been masked (see example names in responses below).

Example run:

	Prompt	Response
1.	Select ADCP File:	9991wh000.cdf
2.	Select Mask File:	9991wh000.msk
3.	Save As ADCP File:	9991whM.cdf

Program:	trimbins.m
Purpose:	trimbins removes data in bins that correspond to layers above the mean sea level. Trimming is performed for data from upward-oriented deployments only. A range to boundary time series is estimated for all ADCPs. Bin distance/height is adjusted for up- or down-looking orientations.
Command:	[MSL, Dstd] = trimbins(numRawFile, ‘rawdata1’, ‘rawdata2’, ‘trimFile’, MSL, Dstd,percentwc)
Description:	trimbins.m is used to modify the data from upward-oriented ADCP deployments so that the remaining bins are at depths below the mean sea level (MSL) plus a standard deviation (Dstd). The mean sea level and the standard deviation may be input; alternatively they are calculated based on, in order of preference, the pressure sensor (pressurecalcs.m), a MATLAB-based range-to-boundary estimator findsurface.m, or by asking the user. When trimbins is run as part of runadcp, MSL and Dstd are derived from the pressure, range, or user input, and the bins will be trimmed accordingly. trimbins removes the same bins from every ensemble. If there was no pressure sensor installed and you wish to override the findsurface program, the total depth and standard deviation can be provided in the command line for trimbins. Note that the total depth refers to the entire water column, including the part below the transducer height. The standard deviation can be approximated as half the tidal range, at the mooring location. When using findsurface.m, trimbins.m creates the brange variable containing the range to boundary estimates generated by findsurface.m.

- If the user suspects that the pressure sensor did not work properly, as to biofouling or other reasons, use findsurface.m or input MSL and Dst manually when using trimbins. This requires running trimbins independently of runadcp.m.
- If this program is run in runadcp.m and there is a pressure sensor installed, there will be no prompts asking for any user information. Therefore, if you believe the pressure sensor has failed, it is important to run trimbins.m independently of runadcp.m and choose a different method for trimming.
- Because data within 6 percent of the surface are contaminated by the surface reflection of the side lobe of the acoustic beam, the user may wish to keep only 94 percent of the water column. However, some investigators believe that useful information can be obtained from bins closer to the surface. This is for the user to decide. (Reference: Principles of Operation: A Practical Primer [for ADCPs]; Teledyne RD Instruments, 2006).
- In this program a new file is not created, but the trimmed ADCP file is simply modified. The existence of a *T.cdf file does not mean that trimbins has been run. If trimbins is run successfully, then it will be reflected in the history attribute of the NetCDF file.

Example run:

	Prompt	Response
1.	How many binary files used?	1
2.	Select Binary ADCP File:	9991wh000.000
3.	Select Ensemble trimmed ADCP File:	9991whT.cdf
4.	Which method to trim the bins?	
	4-1 if pressure sensor installed (proceed to prompt 9)	'Pressure Sensor'
	4-2 if you wish to use your own values (proceed to prompt 8)	'User Input'
5.	A window (see Figure 3) appears displaying the depths calculated by surface for each ensemble:	
	Check surface program output:	
	Minimum depth in meters	17.35
	Maximum depth in meters	18.62
6.	User must input the water depth information	
	mean sea level	
	value	30
	units	meters
	half the tidal range	
	value	1.5
	units	meters
7.	Percent of water column to save	105

Notes on prompts:

Prompt 1. This refers to the number of binary files that were converted to NetCDF format in rdi2cdf.m.

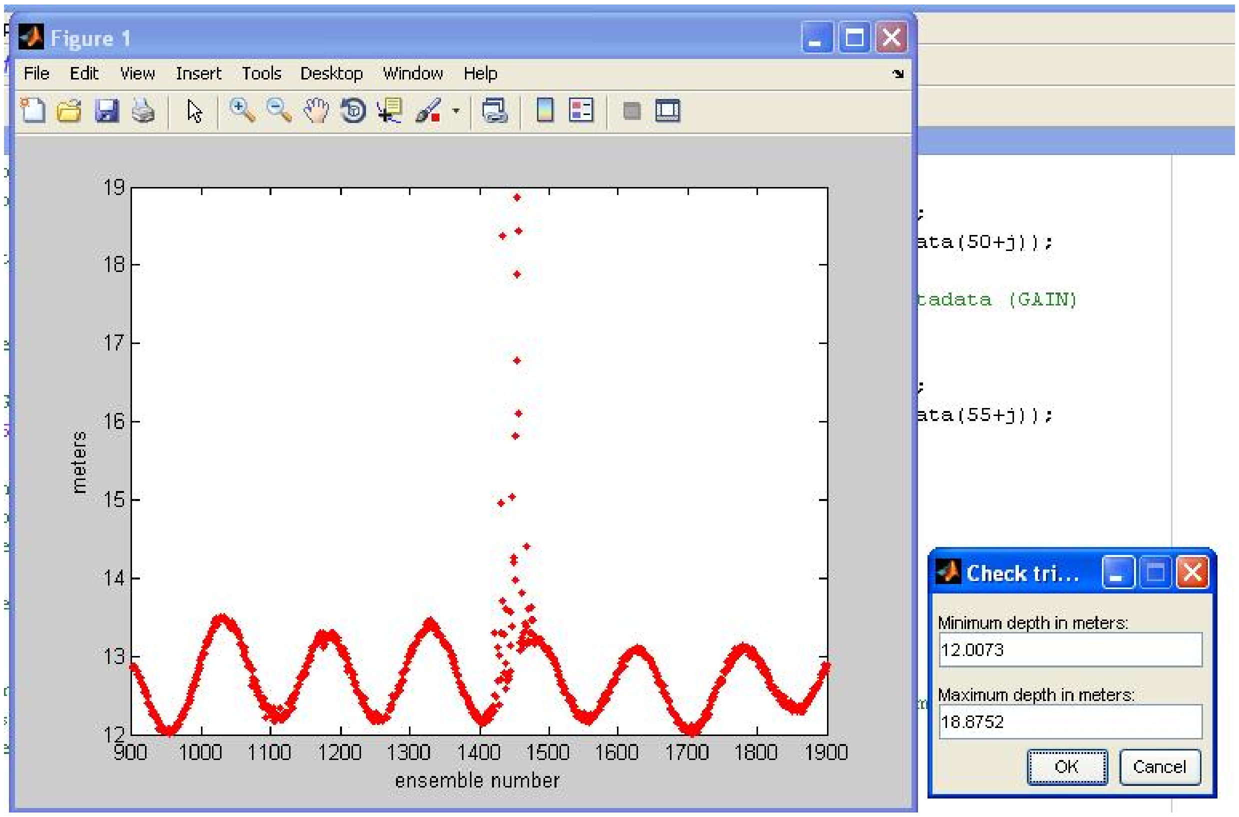
Prompt 4. Choose the method to trim the bins. If 'Pressure Sensor' is chosen and there is no pressure sensor installed, an error message will appear and the program will prematurely terminate.

Prompt 5. An opportunity to change the minimum and maximum depths.

Prompt 6. The TRDI surface program is a DOS-based routine (only runs on PCs) that finds the water depth based on echo intensity. If the user does not have this program, the user must be prepared to provide a depth and half the tidal range for the mooring location.

NOTE: If you have specified the text "pressure sensor" in the "settings" structure, and no pressure sensor is present, you will be asked to choose another method. If you are using a script to process the data in non-interactive mode, the program will stop and will wait for input.

Dependents: pressurecalcs.m, findsurface.m, history.m



```
;
ata(50+j));

tadata (GAIN)

;
ata(55+j));
```

Minimum depth in meters:
12.0073

Maximum depth in meters:
18.8752

Figure 3. Tripod depth verification window.

Program: findsurface.m

Purpose: This program finds the range to a boundary (usually the surface for upward-looking installations) using acoustic backscatter information in the AGC variable.

Command: data = findsurface(‘cdfFile’, ensembles, S, D, trimrange, verbose, allout)

Description: findsurface.m is always called by trimbins to generate the range to boundary variable brange. brange may or may not be used by trimbins to estimate mean sea level, depending on the user's preference. The findsurface program looks at the echo intensity in the bins of each beam for each ensemble to determine the locations of the water surface. Findsurface.m subsamples the output of the surface program to extract the depths only for the ensembles that have been determined to be of good quality. findsurface outputs all of the good range estimates (data.Range2Boundary), the average of the good depths as the mean sea level (MSL) corrected for ADCP height above the bed, and the tidal range divided by two (Dstd), the standard deviation of the range to boundary estimates. Not all data will work well with findsurface. For instance, if enough bins were recorded that the surface reflection and a secondary reflection are present in the RSSI data, findsurface will find the secondary reflection because it looks for a peak starting from the farthest bin and working back toward the ADCP head. In this case, one will have to use the parameter binshift to move the search area for the peak RSSI toward the ADCP head.

Inputs cdfFile - raw NetCDF data file with AGC data

ensembles - ensemble number range, [1 n] or [ ] or for all ensembles

S = salinity to assume for sound speed correction (preferably what was set in the ADCP at deployment), if set to empty matrix ([ ]), the nominal value of 32 will be assumed.

D = depth in meters to use for sound speed correction; this will override a pressure sensor if present.

trimrange - [minDepth maxDepth] | []; determines how to trim outliers; surface detect is often noisy and needs defined limits for best results. Use [ ] to prevent trimming of outliers.

verbose - turn on the plotting feature and watch live. This will seriously slow things down and is used for diagnostics.

binshift – number of bins to shift closer to the ADCP head to look for the peak in acoustic backscatter energy (RSSI)

allout = 1 - pass out all data used, temperature, ensemble number, and such for diagnostic purposes or stand-alone use of this function. This will add the following outputs:

data.ensembles = vector of ensemble numbers (cdf record)

data.peakfit = the peak in AGC determined by parabolic fit for each beam

data.WMVpeak = peak in water mass volume as computed using S, D, and so on

data.pitch = pitch data

data.roll = roll data

data.press = pressure data if present in ADCP

data.Tx = temperature data

data.S = salinity used
data.SVelADCP = sound velocity computed internally by the ADCP

Example run: findsurface.m is not interactive.

Program:	adcp2ep.m
Level:	primary program
Purpose:	This program is used to translate ADCP data into variables relative to earth coordinates and create an EPIC NetCDF data file.
Command:	epDataFile = adcp2ep('adcpFile', 'epDataFile', 'ADCPtype', 'dlgFile').
Description:	adcp2ep is the next program in the ADCP data processing system to be run after the ADCP data are processed by runadcp.m (or the individual programs that comprise the data quality assessment and editing). This program can be run on both beam and earth coordinate data files. Beam coordinate data are transformed into earth coordinates by runbm2g.m, which calls bm2geo.m to do the real work. Earth coordinate data are simply read directly and the variables are renamed from numbered velocities to u, v, w, and so on. The Earth coordinate data will be rotated according to the value in magnetic_variation_applied.

- In the EPIC NetCDF data file, the echo intensity, correlation, and percent good variables are averaged for all 4 beams to produce a single variable for each parameter. Additional processing steps, such as converting the time coordinate variable "TIM" to time and time2, shifting the time base to correspond to the center of the ensemble, and adjusting the depth to be the true water depth of the center of the bin, are handled in this program. Likewise, the units of velocity are converted from millimeters per second to centimeters per second.
- If adcp2ep.m is being run in sequence after runadcp.m, then adcpFile = trimFile in the command line. The 'ADCPtype' input is used to distinguish data from an ADCP "workhorse" from an ADCP "broadband" instrument. This is discussed further in Section 4. Display and Analysis Programs.
- The most time consuming portion of this program is the bm2geo conversion, where the data are rotated into Earth coordinates. A data file cur.mat is created each time this portion of the program is run. In the event that the program crashes before the epDataFile is produced but after the data are transformed, cur.mat can be read in at line 129 of adcp2ep.m.

Example run:

	Prompt		Response
1.	Select ADCP File:		9991whM.cdf
2.	Save data in geographic coordinates as:		9991wh.nc
3.	Select ADCP Deployment Log File:		9991wh.dlg
		OR	9991wh.log
4.	Data Collection Information		
	A. Experiment		'Mass Bay Longterm'
	B. Descript		'Boston Buoy B ADCP'
	C. Project		'MWRA'
	D. Comments		additional data info
	E. Origin		'USGS Woods Hole'
	F. SciPi		'Joe Scientist'

G. Convention	‘EPIC/PMEL’
H. magnetic_variation_applied	0
I. magnetic_variation_at_site	0

Notes on prompts:

Prompt 1. The ADCP file provided should have been processed using runadcp.m, and bad data points removed. According to the format followed in runadcp.m, the file will have a name in the format *T.cdf (the trim file).

Prompt 2. The processed file resulting from this program is typically given the *.nc extension to distinguish it from the other NetCDF data files.

Prompt 3. The deployment log file is a product of the TRDI deployment program. When an ADCP is programmed and “deployed” by the computer, the beam elevations and azimuths specific to that instrument will be recorded in this file.

Prompt 4. A. This is simply the common name of the experiment.

B. Descript is a description of the type of data that are in the NetCDF file. In this case it will always be *something* ADCP.

C. The Project usually describes the main funding agency or field office.

D. Any other comments about the data should be included here.

E. Office that collected the data

F. Chief scientist of the project or principal investigator

G. netCDF convention (USGS follows the EPIC convention from NOAA's Pacific Marine Environmental Lab)

H. This value rotates the heading and current velocity vectors. Degrees west is negative, and the value is added to the heading.

I. This value is only saved as metadata and is not applied to the heading or velocity vectors.

Dependent programs: runbm2g.m, ep_time.m, gregorian.m, uv_rotate.m, history.m

Program:	runbm2g.m
Purpose:	This program gathers the information and reformats the structure of the data to prepare for the transformation from beam to earth coordinates.
Command:	cur = runbm2g('BeamFile', 'ADCPtype', 'dlgFile')
Description:	Runbm2g.m provides the interface between adcp2ep.m and bm2geo.m for data in beam coordinates. This program extracts the compass heading, the pitch, and the roll information from the input beam coordinate file to apply corrections for orientation of the instrument. It also reads the elevations and azimuths for the ADCP instrument from the deployment log file to be used in the transformation matrix. The orientation of the instrument (up or down), as well as the blanking distance, is also obtained from the attributes of the input NetCDF file. After gathering this information, runbm2g.m calls bm2geo.m to convert the data into geographic coordinates ensemble by ensemble. A cell matrix of currents is output by this program as well as saved in a file called cur.mat in the working directory.

- If running this program at the command line, the ADCP type must be specified as BB (broadband) or WH (Workhorse). In interactive format the program will assume that the data are from the Workhorse instrument. More information is given in Section 4 on processing data from a TRDI broadband model ADCP. See below in notes on runbm2g for list of optional command line inputs.
- The primary purpose of runbm2g.m is to read the data from the file into bm2geo.m in the proper format. bm2geo.m can be run independently but it is not recommended (see help bm2geo.m in MATLAB for more information). Other optional inputs for runbm2g.m that are used in running bm2geo.m are seen in bold in the following command line. cur = runbm2g('BeamFile', 'ADCPtype', 'dlgFile', theElevations, theAzimuths, theHeading, thePitch, theRoll, theOrientation, theBlankingDistance)
- There are four beam elevations and azimuths hard-wired into the ADCP instrument. The heading, pitch, and roll data are collected by the ADCP in a time series throughout the deployment. The orientation can be upward- (Up) or downward-looking (down). The blanking distance is the distance above that head of the ADCP where the acoustic signal is contaminated by ringing. This parameter is dictated by the firmware loaded into the instrument.

Example run:

	Prompt		Response
1.	Select ADCP File in Beam coordinates:		9991whT.cdf
2.	Select ADCP Deployment Log File:		9991wh.dlg
		OR	9991wh.log

Notes on prompts:

Prompt 1. The input ADCP file should be devoid of bad data points as a result of being passed through the filtering routines of runadcp.m. According to the format followed in runadcp.m, the file will have a name in the format *T.cdf (the trim file).

Prompt 2. The deployment log file is a product of the TRDI deployment program. When an ADCP is programmed and “deployed” by the computer, the beam elevations and azimuths specific to that instrument will be recorded in this file.

Dependent program: bm2geo.m

Program:	pressuremask.m
Purpose:	This program uses pressure or height data in the NetCDF files to mask data that are above the surface, following the tidally varying surface level. This program can also use external data input by the user.
Command:	[theNewADCPFile, theMaskFile] = pressuremask (‘theDataFile’,[‘theMaskFile’],[‘theNewADCPFile’], [thePad], [heightData])
Description:	Pressuremask.m can be run independently, if desired, and is called with the above command statement. It is not part of the runadcp process. If the raw NetCDF file name (theDataFile), is not specified in the call statement the user will be prompted by dialog boxes. Pressuremask.m creates a mask file (theMaskFile) which is identical to the raw NetCDF file in size and name with a *.msk extension and fills the variables with 0's. If it finds an existing mask file, it will alter this mask file. The variables in the raw NetCDF file are then scanned to find data that are above the sea surface based on the pressure or height variables. Pressuremask.m then calls postmask.m to apply the mask file (of logical indices) to the rawcdf file producing theNewADCPFile, which is suggested to have a name like *P.cdf. If this function was run successfully it will produce theMaskFile (*.msk) file and the theNewADCPFile (*P.cdf).

- Other inputs: thePad is optional and allows the user to enter a value, in meters, that the program will add to the water depth value used to screen the data. This will result in preservation of data above the sea surface, useful for some analyses. heightData may be provided if the file does not contain pressure or height information. Values must be in meters above the sea bed.
- The pressure mask can be applied to data in earth coordinates. The typical time to use this function is after runadcp has been executed and before adcp2ep is executed.

Example run:

	Prompt	Response
1.	Select NetCDF ADCP File:	9991wh000T.cdf

Program:	fixorientation.m
Purpose:	TRDI ADCPs are able to detect the orientation in which they are deployed. The check, however, happens only once, when the ADCP wakes up and starts logging. Sometimes, the ADCP may not yet be in its deployed orientation, and thus an upward-looking ADCP may have “DOWN” stored in its header. This program fixes this problem.
Command:	fixorientation(‘filename’, ‘new_orientation’)
Description:	To use fixorientation, run rdi2cdf separately, before using runadcp. Then run fixorientation, giving the name of rdi2cdf’s output file. The orientation flag will be changed and the depths recalculated to reflect the new orientation. At this point, runadcp can be executed and told to use the output file generated by rdi2cdf and fixed by fixorientation.

Program:	pressurecalcs.m
Purpose:	Determines the mean sea level and the tidal fluctuation derived from the raw pressure sensor data. Writes a csv file with *.dat extension containing the depth at each ensemble.
Command:	[MSL, Dstd, Dout] = pressurecalcs(‘adcpFile’)
Description:	Input a raw ADCP data file (*.cdf) with pressure data to get MSL (mean sea level), Dstd, the standard deviation of the pressure for an approximation of the tidal excursion, Dout, the surface depth at each ensemble.

Program:	checktime.m
Purpose:	This program provides the user statistics on the performance of the ADCP clock, to assist in identifying gaps, restarts, and so on.
Command:	checktime(‘cdfFile’, TE, verbose)
Description:	Operates on *.cdf or *.nc (EPIC) files. cdfFile is the NetCDF data file. TE is the ensemble interval which can be found in the *.whp file output by TRDI’s setup software or can be found by inspecting the raw binary data (*.000 or *.PD0) file using TRDI winadcp’s Utilities -> File Details option. Setting verbose to 0 suppresses the histogram plot output. The default is 1.

Program:	splitadcp.m
Purpose:	This program separates wave packet data from current velocity ensemble data; the current velocity ensembles are saved in a separate binary file ready for processing by this toolbox.
Command:	splitadcp (‘rawADCP file’, [savepackets])
Description:	Operates on raw binary ADCP data. User may set "savepackets" to "1" to output the wave packet data separately; however, these packets-only data cannot be read by wavesmon.

Section 4. Display and Analysis Programs

<u>ncBrowse</u>

NcBrowse is a very good viewer which can plot and export the contents of a *.nc or *.cdf file. NcBrowse is available at http://www.epic.noaa.gov/java/ncBrowse/. NcBrowse does provide the functionality of maskADCP, which offers a basic point and click environment to allow users to set bad data flags in a masking file that is used to replace data values with fill values. Users familiar with accessing NetCDF files directly from MATLAB may prefer to write their own scripts to mask and edit the NetCDF files generated by this toolbox.

<u>Interactive masking</u>

Masking data interactively can be performed by simply typing runmask at the MATLAB >> prompt. runmask, when no arguments are supplied, will run in interactive mode and allow the user to do additional data editing via a GUI window. See maskadcp.m for details.

<u>Time checks</u>

The program checktime.m gives various statistics about the time stamps in the data file.

Section 5. Tips and Tools

<u>Instrument setup</u>

The user should remember the following when setting up the ADCP instrument prior to deployment that will enhance the use of this processing system.

1. If the instrument is deployed on a fixed platform, it is recommended that data be recorded in beam coordinates. This is not desirable for instruments mounted on moving platforms.
2. The user should note if a heading bias was given to the ADCP prior to deployment. When requested for the magnetic declination during runadcp.m, it is recommended that the value 0 be entered since the heading will be corrected by that value in the ADCP during data collection and, if care is not used, the data would be rotated again by the post-processing software.
3. Save the deployment log file generated by the TRDI deploy program. If the deployment log file is not available, the azimuths and elevations for the specific ADCP must be obtained through communication with the instrument using the PS3 command. The default settings for the elevations and azimuths may be used as well, see Broadband Data below.

<u>Demo data sets</u>

There are three example data sets in subdirectories called Demo, Demo2, and Demo3.

Demo: Demo is a very simple, short data set that was recorded by a non-waves array, bottom-mounted, up-looking instrument. Data were recorded in beam coordinates.

Demo2: Demo2 was recorded by a bottom-mounted, up-looking instrument that also recorded waves, and the *.PD0 files are the binary output of splitadcp.m. Demo2 is an example of how the USGS Woods Hole Science Center uses this toolbox to process ADCP data. Demo2 includes an extra Microsoft Word file and mfile which record and handle metadata. This data set also has a bad profile at index 14136 (TRDI ensemble number 25706). This bad profile only gets masked in the beam 2 data and subsequently gets propagated in the final .nc file. This masking behavior will be improved in a future release, but maskADCP.m may be used to correct this. A script file "howtoplotdemo2.m" demonstrates the use of "autonan" to load and plot data. The demo also illustrates the effects of a bad profile, such as in this example, on plotting in MATLAB.

Demo3: Demo3 is data from an instrument mounted down-looking on a subsurface mooring where the sea bed was in range of the instrument. Data were recorded in earth coordinates. This instrument also recorded bottom track data, and the bottom track data can be found in the *.cdf files but are not saved to the *.nc files by the toolbox. This may change in a future version.

All the demo data sets contain the script files that execute this toolbox and provide a good way to learn how to automate processing with this toolbox. These data sets are also used to test and troubleshoot the ADCP toolbox code.

Broadband Data

Runadcp.m will process data from a TRDI Broadband model ADCP without any difficulty. The only modification that needs to be made is in adcp2ep.m. For broadband data the ADCP type must be specified to be Broadband in adcp2ep.m, otherwise the program will assume the data are from a workhorse. This is accomplished by using the call statement: adcp2ep('adcpFile', 'epDataFile', 'ADCPtype'), specifying the input and output file names and setting ADCPtype to BB. Since the broadband instrument has a perfect beam configuration, the defaults are used; the elevations are all –70 and the azimuths are exactly 270, 90, 0, and 180.

Time Savers

The amount of time that it takes to process an ADCP data set can be reduced by the following methods.

Runadcp/Rdi2cdf

The most time consuming portion of runadcp is the initial conversion of the data from binary to NetCDF format. This conversion actually occurs in the rdi2cdf function. Rdi2cdf can be run independently of runadcp and only requires user interaction at the very beginning of the program. See the description of the Rdi2cdf.m program in Section 2 for details on running this program. Once the initial NetCDF is produced, the file can be used in runadcp to perform the data quality checking steps. The first two input filenames for runadcp must be specified at the command line in order to use the program in this manner.

Automated Processing
Automated processing was developed so that the ADCP data processing programs could be run with minimal user intervention. Parameters are input directly using data structures in a MATLAB script file, and browsers for inspection of the data are suppressed. For problematic data files with large numbers of bad data points, automated processing may be very useful to process subsets of data and try different options. The file scriptexample.m can be used to implement automated processing. The comments and structure data field names will guide the user for file names, metadata, settings, and functions. A good knowledge of the toolbox is required.

Making your own masking program

Pressuremask.m is an example of how a specialized mask can be quickly constructed and applied to ADCP data processed by this toolbox. It can be done with raw (*.cdf) and fully processed (*.nc) data. The steps are as follows:

1. Use mkadcpmask.m to create a "blank" mask file based on your data file. You do not need any information about the data except to know that it is an ADCP data file converted by rdi2cdf. Invocation is as follows: mkadcpmask('inFile', 'outFile', fillValue). Mkadcpmask.m is part of the this toolbox.
2. Change the mask based on your criteria for screening the data. All the data in the new mask (*.msk) file will be zero but will have the same shape and same variable names as the data file from which it was modeled. Any data point set to 1 will be interpreted in the next step as a bad data flag and will cause the corresponding value (variable, time, beam, and bin) in the ADCP data set to be replaced with the fill value.
3. Apply the mask using the postmask.m program, invoked by: postmask('theDataFile', 'theMaskFile', 'theNewADCPFile'). Postmask writes a new file based on the mask and does not touch the original data file.

In the standard data processing sequence, maskADCP and fillmsk both use this technique to edit ADCP data.

Section 6. Troubleshooting

Encountering Errors

If runadcp.m or adcp2ep.m stops before completion as the result of an error, the user may be able to begin processing where the program left off. This is the value of keeping a diary file of output to the command window. Common reasons for failure are:
- Not all subdirectories of the ADCP toolbox are on the MATLAB path
- The NetCDF and mexnc toolboxes are missing or improperly installed
- A data file name is inaccurate or the data file cannot be found on disk

The History attribute in the NetCDF file contains a log of the programs that have been successfully run on the data file. The comments are in reverse chronological order such that the program most recently completed appears first. If runadcp or adcp2ep has not been completed, then the user may use the history information to run the failed processing programs independently.

The History attribute for the Best Basic Version of ADCP data collected in beam coordinates, with missing ensembles, should appear as follows:

> Written to an EPIC standard data file by adcp2ep.m (version 1.1);
> Transformed to earth coordinates by runbm2g.m;
> Bins were trimmed by trimbins.m based on (findsurface.m, Pressure sensor, user input);
> The data were filtered using TRDI quality control factors in runmask.m;
> The missing ensemble numbers that were filled with fill values using fixEns.m were: 32; 467; 1029;
> Converted to NetCDF via MATLAB by rdi2cdf.m 3.1 14-Dec-2004

For data collected in earth coordinates with missing ensembles, the History attribute will appear as:

> Written to an EPIC standard data file by adcp2ep.m (version 1.1);
> Bins were trimmed by trimbins.m based on findsurface.m output;
> The missing ensemble numbers that were filled with fill values using fixEns.m were: 32; 467; 1029;
> Converted to NetCDF via MATLAB by rdi2cdf.m 3.1 14-Dec-2004

Ensemble trimming

Occasionally, bad ensembles may remain at the ends of the data record. The nctrim utility from the NetCDF toolbox can be used to remove ensembles from the ends of the record. The nctrim utility can be used at any stage of processing. The command line statement is nctrim('theSrc','theDst',theIndices,'theRecDim'), where theSrc is the NetCDF file that needs to be trimmed and the resulting file will be theDst. The indices can be given by a range such as 100:3000, which removes everything outside of that range. Be careful not to confuse the ensemble (record) number, also indicated as the 'Rec' variable in the NetCDF file, with the index number. The ensemble numbers start at 1 in the raw NetCDF file. However, as the data are processed, bad ensembles may be removed. Therefore, the ensemble number may not match the index number in the data file. It is the index numbers that must be given in nctrim.

Severely tilted ADCPs

Sometimes, when an ADCP is set up to be on a fixed platform, like a benthic tripod, it is discovered upon recovery to have been severely tilted during some or all of the deployment. This presents a problem when trimming the bins that are at or above the sea surface, using the conventional program trimbins.m. If the tilt sensor indicates a value greater than 10 degrees in either pitch or roll, it is recommended that tilttrim.m is run instead of trimbins.m. Although tilttrim.m may take very long to run, it may salvage a few bins of data that would otherwise be discarded. Three conditions should be met for tilttrim.m to be run: the data were recorded in beam coordinates, the unit had significant tilt, and the instrument was upward-looking.

ADCPs that can't see the surface
When an ADCP is deployed in a location that is deeper than its maximum profiling range, it will not detect the surface. In this case, the automatic processes the toolbox uses to determine the mean sea level will be incorrect and the depth values will be incorrect. In this case, "User Input" method should be use for trimming the bins and the mean sea level at the site, and tidal range must be manually entered.

Section 8. Known Issues

Working with two binary files

The NetCDF toolbox can take two binary files as inputs. However, rdi2cdf.m can crash if there is an incomplete record at the end of the first binary file that it cannot detect. If this happens, concatenate the two binary files using fappend.m.

Winriver II output files

TRDI publishes software for index velocity measurements called winriver II. This software package outputs a PD0 format file that is not quite the same as PD0 format stored by Workhorse ADCPs during self-contained deployments. Unfortunately, winriver II output files are not compatible with this software package. Other USGS software has been developed for this application and may be found at http://hydroacoustics.usgs.gov/.

Gaps in ensemble numbers

Large gaps in ensemble numbers will cause this software package to crash. Fixensemble.m is constrained by MATLAB's interp1.m function and will not handle large gaps well.

Section 9. Selected References

Denbo, D. W., 2006, ncBrowse, A Graphical NetCDF File Browser, version 1.6.3: National Oceanographic and Atmospheric Administration Pacific Marine Environmental Laboratory, available online at http://www.epic.noaa.gov/java/ncBrowse/ (Accessed July 23, 2009).

Evans, John, 2008, MEXNC, SNCTOOLS, and the NetCDF Toolbox, version 2.9.10, available online at http://mexcdf.sourceforge.net/ (Accessed July 23, 2009).

National Oceanographic and Atmospheric Administration Pacific Marine Environmental Laboratory, 2006, EPIC, available online at http://www.epic.noaa.gov/epic/ (Accessed July 23, 2009).

Teledyne RD Instruments, 2005, WorkHorse Monitor, Sentinel, Mariner, Rio Grande, Quartermaster, H-ADCP, Navigator, and Long Ranger ADCPs -- Commands and output data format: P/N 957-6156-00, 206 p.

Teledyne RD Instruments, 2006, Acoustic Doppler Current Profiler principles of operation -- A practical primer: P/N 951-6069-00, 59 p.